Guide du développeur pour la version 3

AWS SDK pour PHP



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK pour PHP: Guide du développeur pour la version 3

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que le AWS SDK pour PHP ? 1
Commencez avec le SDK 1
Ressources supplémentaires 1
Documentation sur les API 2
Maintenance et prise en charge des versions majeures du kit SDK 2
Premiers pas 3
Conditions préalables 3
Exigences 3
Recommandations 3
Test de compatibilité 5
Installation 5
Installation AWS SDK pour PHP en tant que dépendance via Composer 5
Installation à l'aide du package phar 7
Installation à l'aide du fichier ZIP 7
Authentification avec AWS 8
Configurer l'authentification avec IAM Identity Center pour le développement local 8
Démarrez un AWS session du portail d'accès 9
En savoir plus sur l'authentification 10
Création d'une application simple 10
Prérequis 10
Inclure le SDK dans votre code 11
Ecriture du code 11
Exécution du programme 12
Étapes suivantes 12
Utilisation AWS Cloud9 avec le SDK 12
Étape 1 : Configurez votre appareil Compte AWS pour l'utiliser AWS Cloud9 13
Étape 2 : Configuration de votre environnement AWS Cloud9 de développement 13
Étape 3 : Configurez le AWS SDK pour PHP 14
Étape 4 : Téléchargez un exemple de code 15
Étape 5 : Exécuter un exemple de code 16
Configuration des clients de service 18
Configuration du client en externe 18
Chaîne de fournisseurs de configuration pour la configuration du client 19
Création d'un client de service configuré à l'aide de paramètres externes 19

AWS SDK pour PHP Variables d'environnement version 3 20
Configuration du client dans le code 21
Configuration de base dans le code 21
Utilisation de la Sdk classe 22
Options du constructeur 23
Région AWS 53
Chaîne de résolution régionale 53
Bonnes pratiques 54
Fournisseurs d'informations d'identification 55
Qu'est-ce qu'un fournisseur d'informations d'identification dans la AWS SDK pour PHP version 3 ? 55
Comprendre la chaîne de fournisseurs d'informations d'identification par défaut dans la AWS SDK pour PHP version 3 56
Fournisseurs d'informations d'identification intégrés dans la AWS SDK pour PHP version 3 57
Enchaînement des fournisseurs d'identifiants dans le SDK for PHP 69
Création d'un fournisseur d'informations d'identification personnalisé à utiliser avec le SDK for PHP 70
Mémorisation des informations d'identification dans le SDK for PHP 71
Assumez un rôle IAM 72
Utiliser des informations d'identification temporaires depuis AWS STS le SDK for PHP 79
Créez des clients anonymes dans le SDK for PHP 82
Utilisez l'extension AWS CRT 82
Ai-je besoin de l'extension AWS CRT ? 82
Comment installer l'extension AWS CRT ? 83
Utilisation de l’SDK 84
Faire des Service AWS demandes 84
Vue d'ensemble du flux de travail des demandes de SDK 84
Création d'un client de service de base 85
Demandes 86
Utilisation des objets de résultat 87
Objets de commande 89
Programmation asynchrone 96
Promesses 98
Gestion des erreurs 105
Gestion des erreurs synchrones 105

Gestion asynchrone des erreurs 106
Gestionnaires et intergiciels 108
Gestionnaires 108
Intergiciel 110
Création de gestionnaires personnalisés 118
Streams 119
Décorateurs de stream 119
Pagination 123
Objets du paginateur 124
Énumération des données à partir des résultats 125
Pagination asynchrone 125
Programmes d'attente 127
Configuration du serveur 127
En attente asynchrone 129
JMESPath expressions 130
Extraction de données à partir de résultats 131
Extraction de données à partir de paginateurs 135
Appel Services AWS 137
Fonctionnalités et options d'utilisation 137
Amazon DynamoDB 137
Amazon S3 145
Exemples de code avec conseils 232
Informations d’identification 232
CloudFront Exemples Amazon 233
Amazon CloudSearch 262
CloudWatch Exemples Amazon 264
EC2 Exemples Amazon 289
Amazon OpenSearch Service 302
Gestion des identités et des accès AWS exemples 304
AWS Key Management Service 329
Exemples Kinesis 351
AWS Elemental MediaConvert 368
Exemples d'Amazon S3 375
AWS Secrets Manager 410
Exemples Amazon SES 420
Exemples d'Amazon SNS 453

Exemples Amazon SQS 472
Amazon EventBridge 485
Exemples de code 487
API Gateway 488
Actions 488
Scénarios 494
Aurora 494
Scénarios 494
Auto Scaling 495
Prise en main 496
Principes de base 497
Actions 488
Amazon Bedrock 511
Actions 488
Exécution d’Amazon Bedrock 513
Amazon Nova 513
Stable Image Core 515
Amazon DocumentDB 516
Exemples sans serveur 516
DynamoDB 518
Principes de base 497
Actions 488
Scénarios 494
Exemples sans serveur 516
Amazon EC2 551
Actions 488
AWS Glue 557
Principes de base 497
Actions 488
IAM 577
Principes de base 497
Actions 488
Kinesis 595
Exemples sans serveur 516
AWS KMS 598
Prise en main 496

Principes de base 497
Actions 488
Lambda 635
Principes de base 497
Actions 488
Scénarios 494
Exemples sans serveur 516
Amazon MSK 665
Exemples sans serveur 516
Amazon RDS 667
Actions 488
Scénarios 494
Exemples sans serveur 516
Amazon RDS Data Service 676
Scénarios 494
Amazon Rekognition 677
Scénarios 494
Amazon S3 678
Mise en route 496
Principes de base 497
Actions 488
Scénarios 494
Exemples sans serveur 516
Compartiments de répertoires S3 702
Principes de base 497
Amazon SES 718
Scénarios 494
Amazon SNS 719
Actions 488
Scénarios 494
Exemples sans serveur 516
Amazon SQS 740
Exemples sans serveur 516
Migration depuis la version 2 744
Introduction 744
Quoi de neuf dans la version 3 ? 744

Couche HTTP découplée 744
Requêtes asynchrones 96
Qu'est-ce qui est différent de la version 2 ? 745
Les dépendances du projet sont mises à jour 745
Les options de région et de version sont désormais obligatoires 746
L'instanciation du client utilise le constructeur 746
La configuration du client a changé 747
Certains résultats de l'API ont changé 748
Les classes Enum ont été supprimées 751
Les classes d'exception détaillées ont été supprimées 751
Les classes de façade statiques ont été supprimées 752
Les paginateurs remplacent les itérateurs 752
De nombreuses abstractions de haut niveau ont changé 753
Comparaison d'échantillons de code provenant des deux versions du SDK 754
Exemple : `ListObjects` opération Amazon S3 754
Exemple : Instanciation d'un client avec une configuration globale 756
Sécurité 758
Protection des données 758
Gestion de l’identité et des accès 760
Public ciblé 760
Authentification par des identités 761
Gestion de l’accès à l’aide de politiques 762
Comment ? Services AWS collaborez avec IAM 764
Résolution des problèmes AWS identité et accès 764
Validation de la conformité 766
Résilience 767
Sécurité de l’infrastructure 768
Migration du client de chiffrement Amazon S3 (V1 vers V2) 768
Présentation de la migration 769
Mettre à jour les clients existants pour lire les nouveaux formats 769
Migrer les clients de chiffrement et de déchiffrement vers la version V2 770
Exemples de migration 771
Migration du client de chiffrement Amazon S3 (V2 vers V3) 774
Présentation de la migration 774
Comprendre les concepts de la V3 774
Mettre à jour les clients existants pour lire les nouveaux formats 777

Migrer les clients de chiffrement et de déchiffrement vers la version 3 778
Exemples supplémentaires 783
FAQ 787
Quelles méthodes sont disponibles sur un client ? 787
Que dois-je faire en cas d'erreur de certificat cURL SSL ? 787
Quelles versions d'API sont disponibles pour un client ? 787
Quelles versions de région sont disponibles pour un client ? 788
Pourquoi ne puis-je pas charger ou télécharger des fichiers de plus de 2 Go ? 788
Comment savoir quelles données sont envoyées sur le réseau ? 788
Comment puis-je définir des en-têtes arbitraires sur une requête ? 789
Comment puis-je signer une requête arbitraire ? 789
Comment puis-je modifier une commande avant de l'envoyer ? 789
Qu'est-ce qu'un CredentialsException ? 789
Est-ce que cela AWS SDK pour PHP fonctionne sur HHVM ? 790
Comment désactiver SSL ? 790
Que dois-je faire face à une « erreur d'analyse » ? 791
Pourquoi le client Amazon S3 décompresse-t-il les fichiers compressés au format gzippé ? 791
Comment désactiver la signature corporelle dans Amazon S3 ? 791
Comment le schéma de nouvelle tentative est-il géré dans le kit AWS SDK pour PHP ? 792
Comment gérer les exceptions avec des codes d'erreur ? 792
Glossaire 794
Historique du document 797
..... dcccii

Qu'est-ce que la AWS SDK pour PHP version 3 ?

La AWS SDK pour PHP version 3 permet aux développeurs PHP d'utiliser Amazon Web Services dans leur code PHP et de créer des applications et des logiciels robustes à l'aide de services tels qu'Amazon S3, Amazon DynamoDB et Amazon Glacier. Vous pouvez démarrer en quelques minutes en installant le SDK via Composer (en exigeant le `aws/aws-sdk-php` package) ou en téléchargeant le logiciel autonome `aws.zip`ou `aws.phar`le fichier.

Tous les services ne sont pas immédiatement disponibles dans le kit SDK. Pour savoir quels services sont actuellement pris en charge par le AWS SDK pour PHP, voir Nom du service et version de l'API.

> **Note**
>
> Si vous migrez votre code depuis la version 2 du SDK vers la version 3, assurez-vous de lire Mettre à niveau depuis la version 2 du. AWS SDK pour PHP

Commencez avec le SDK

Si vous êtes prêt à vous familiariser avec le SDK, suivez le Commencer à utiliser la AWS SDK pour PHP version 3 chapitre. Il vous guide tout au long de l'authentification AWS, de la configuration de votre environnement de développement et de la création de votre première application de base à l'aide d'Amazon S3.

Ressources supplémentaires

- FAQ
- Glossaire
- AWS SDKs et Guide de référence des outils : contient des paramètres, des fonctionnalités et d'autres concepts fondamentaux courants entre AWS SDKs.
- Documentation Guzzle
- Des exemples de code utilisant le AWS SDK pour PHP sont disponibles dans le dépôt awsdocs/aws-doc-sdk-examples.
- Communauté du SDK PHP sur Gitter.
- AWS re:Post.

GitHub:

- Le code source de AWS SDK pour PHP est disponible dans le aws-sdk-php dépôt aws/.
- Contribution au kit SDK
- Signaler un bogue ou demander une fonctionnalité

Documentation sur les API

Retrouvez la documentation de l'API pour le SDK sur https://docs.aws.amazon.com/sdk-for-php/latest/reference/.

Maintenance et prise en charge des versions majeures du kit SDK

Pour plus d'informations sur la maintenance et le support des versions majeures du SDK et de leurs dépendances sous-jacentes, consultez les informations suivantes dans le guide de référence AWS SDKs and Tools :

- AWS SDKs et politique de maintenance des outils
- AWS SDKs et matrice de support des versions d'outils

Commencer à utiliser la AWS SDK pour PHP version 3

Ce chapitre est dédié à vous aider à démarrer avec la AWS SDK pour PHP version 3.

Rubriques

- Exigences et recommandations pour la AWS SDK pour PHP version 3
- Installation de la AWS SDK pour PHP version 3
- Authentification avec AWS utilisant AWS SDK pour PHP Version 3
- Création d'une application simple à l'aide de la AWS SDK pour PHP version 3
- À utiliser AWS Cloud9 avec la AWS SDK pour PHP version 3

Exigences et recommandations pour la AWS SDK pour PHP version 3

Pour de meilleurs résultats AWS SDK pour PHP, assurez-vous que votre environnement répond aux exigences et recommandations suivantes.

Exigences

Pour utiliser le AWS SDK pour PHP, vous devez utiliser PHP version 8.1 ou ultérieure avec l'extension PHP SimpleXML activée. Si vous devez signer un compte Amazon privé CloudFront URLs, vous avez également besoin de l'extension PHP OpenSSL.

Recommandations

En plus de la configuration minimale requise, nous vous recommandons d'installer, de désinstaller et d'utiliser les éléments suivants.

Installez cURL 7.16.2 ou une version ultérieure	Utilisez une version récente de cURL compilée avec OpenSSL/NSS et zlib. Si cURL n'est pas installé sur votre système et si vous ne configurez pas de gestionnaire HTTP personnal isé pour votre client, le kit SDK utilisera l'encapsuleur de flux PHP.

Utilisation des OPCache	Utilisez l' OPcache extension pour améliorer les performances de PHP en stockant le bytecode du script précompilé dans la mémoire partagée. Ainsi, vous n'avez plus besoin de PHP pour charger et analyser des scripts sur chaque requête. Cette extension est généralement activée par défaut. Lorsque vous exécutez Amazon Linux, vous devez installer le package php56-opcache ou php55-opcache yum pour utiliser l'extension. OPCache
Désinstallez Xdebug dans les environnements de production	Xdebug peut vous aider à identifier les goulots d'étranglement au niveau des performances. Toutefois, si les performances sont essentielles pour votre application, n'installez pas l'extensi on Xdebug dans votre environnement de production. Le chargement de cette extension ralentit considérablement les performances du kit SDK.
Utilisez un chargeur automatique classmap Composer	Les chargeurs automatiques chargent les classes requises par un script PHP. Composer génère un chargeur automatique capable de charger automatiquement les scripts PHP de votre application ainsi que tous les autres scripts PHP qu'elle requiert, y compris le kit AWS SDK pour PHP. Nous vous recommandons d'utiliser un chargeur automatique classmap pour les environnements de production afin d'améliorer leurs performances. Vous pouvez générer un chargeur automatique classmap en spécifiant l'option `-o` ou `==optimize-autoloader` dans la commande d'installation de Composer.

Test de compatibilité

Exécutez le `compatibility-test.php`fichier situé dans la base de code du SDK pour vérifier que votre système peut exécuter le SDK. En plus de vérifier la configuration système minimale requise pour le kit SDK, le test de compatibilité contrôle certains paramètres facultatifs et propose des recommandations qui peuvent vous aider à améliorer les performances. Le test de compatibilité génère des résultats dans la ligne de commande ou dans un navigateur web. Si vous passez en revue les résultats du test dans un navigateur, les vérifications réussies s'affichent en vert, les avertissements s'affichent en violet et les échecs s'affichent en rouge. Lorsque vous exécutez le test à partir de la ligne de commande, chaque résultat d'un contrôle s'affiche sur une ligne distincte.

Lorsque vous signalez un problème concernant le kit SDK, pensez à partager le résultat du test de compatibilité afin de faciliter l'identification de la cause sous-jacente.

Installation de la AWS SDK pour PHP version 3

Vous pouvez installer la AWS SDK pour PHP version 3 :

- En tant que dépendance via Composer
- En tant que phar prépackagé du kit SDK
- En tant que fichier ZIP du kit SDK

Avant d'installer AWS SDK pour PHP la version 3, assurez-vous que votre environnement utilise PHP version 8.1 ou ultérieure. En savoir plus sur les exigences et les recommandations en matière d'environnement.

> **Note**
>
> L'installation du SDK via les méthodes .phar et .zip nécessite que l'extension PHP Multibyte String soit installée et activée séparément.

Installation AWS SDK pour PHP en tant que dépendance via Composer

Composer est la méthode recommandée pour installer le AWS SDK pour PHP. Composer est un outil PHP qui gère et installe les dépendances de votre projet.

Pour en savoir plus sur l'installation de Composer, la configuration du chargement automatique et le respect d'autres bonnes pratiques pour définir des dépendances, rendez-vous sur getcomposer.org.

Installation de Composer

Si Composer n'est pas déjà dans votre projet, téléchargez et installez Composer sur la page Télécharger Composer.

- Pour Windows, suivez les instructions de Windows Installer.
- Pour Linux, suivez les instructions d'installation en ligne de commande.

Ajouter AWS SDK pour PHP en tant que dépendance via Composer

Si Composer est déjà installé globalement sur votre système, exécutez ce qui suit dans le répertoire de base de votre projet pour l'installer AWS SDK pour PHP en tant que dépendance :

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

Sinon, tapez cette commande Composer pour installer la dernière version du AWS SDK pour PHP en tant que dépendance.

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

Ajouter un chargeur automatique à vos scripts php

L'installation de Composer crée plusieurs dossiers et fichiers dans votre environnement. Le fichier principal que vous utiliserez est `autoload.php`, qui se trouve dans le dossier `vendor` de votre environnement.

Pour utiliser le AWS SDK pour PHP dans vos scripts, incluez le chargeur automatique dans vos scripts, comme suit.

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

Installation à l'aide du package phar

Chaque version AWS SDK pour PHP inclut un phar (archive PHP) préemballé qui contient toutes les classes et dépendances dont vous avez besoin pour exécuter le SDK. De plus, le phar enregistre automatiquement un chargeur automatique de classe pour toutes AWS SDK pour PHP ses dépendances.

Vous pouvez télécharger le phar packagé et l'inclure dans vos scripts.

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

> **Note**
>
> L'utilisation de PHP avec le correctif Suhosin n'est pas recommandée, mais elle est courante sur les distributions Ubuntu et Debian. Dans ce cas, il se peut que vous ayez besoin d'activer l'utilisation des phars dans le suhosin.ini. Si vous ne le faites pas, l'ajout d'un fichier phar à votre code provoquera un échec silencieux. Pour modifier suhosin.ini, ajoutez la ligne suivante.
>
> ```
> suhosin.executor.include.whitelist = phar
> ```

Installation à l'aide du fichier ZIP

AWS SDK pour PHP Il inclut un fichier ZIP contenant toutes les classes et dépendances dont vous avez besoin pour exécuter le SDK. De plus, le fichier ZIP inclut un chargeur automatique de classe pour le kit AWS SDK pour PHP et ses dépendances.

Pour installer le kit SDK, téléchargez le fichier .zip, puis extrayez-le dans votre projet à l'emplacement de votre choix. Incluez ensuite le chargeur automatique dans vos scripts, comme suit.

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

Authentification avec AWS utilisant AWS SDK pour PHP Version 3

Vous devez définir la manière dont votre code s'authentifie AWS lorsque vous développez avec Services AWS. Vous pouvez configurer l'accès programmatique aux AWS ressources de différentes manières en fonction de l'environnement et de l' AWS accès dont vous disposez.

Pour choisir votre méthode d'authentification et la configurer pour le SDK, consultez la section Authentification et accès dans le Guide de référence AWS des SDK et des outils.

Configurer l'authentification avec IAM Identity Center pour le développement local

Nous recommandons aux nouveaux utilisateurs qui se développent localement et qui ne reçoivent aucune méthode d'authentification de la part de leur employeur de le configurer AWS IAM Identity Center. Cette méthode inclut l'installation AWS CLI pour faciliter la configuration et pour vous connecter régulièrement au portail AWS d'accès. Si vous choisissez cette méthode, votre environnement doit contenir les éléments suivants une fois que vous avez terminé la procédure d'authentification IAM Identity Center décrite dans le Guide de référence AWS des SDK et des outils :

- Le AWS CLI, que vous utilisez pour démarrer une session de portail d' AWS accès avant d'exécuter votre application.
- AWS`config`Fichier partagé doté d'un `[default]` profil avec un ensemble de valeurs de configuration pouvant être référencées par le SDK. Pour connaître l'emplacement de ce fichier, consultez Location of the shared files dans le manuel AWS SDKs and Tools Reference Guide.
- Le `config` fichier partagé contient le `region`paramètre. Cela définit la valeur par défaut Région AWS que le SDK utilise pour les demandes. Cette région est utilisée pour les demandes de service du SDK qui ne sont pas explicitement configurées avec une `region` propriété.
- Le SDK utilise la configuration du fournisseur de jetons SSO du profil pour obtenir des informations d'identification avant d'envoyer des demandes à. AWS La `sso_role_name` valeur, qui est un rôle IAM connecté à un ensemble d'autorisations IAM Identity Center, permet d'accéder à l'utilisateur Services AWS dans votre application.

  Le `config` fichier d'exemple suivant montre un profil par défaut configuré avec la configuration du fournisseur de jetons SSO. Le `sso_session` paramètre du profil fait référence à la `sso-session`section nommée. La `sso-session` section contient les paramètres permettant de lancer une session sur le portail AWS d'accès.

```
[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
```

Il AWS SDK pour PHP n'est pas nécessaire d'ajouter des packages supplémentaires (tels que `SSO` et`SSOOIDC`) à votre application pour utiliser l'authentification IAM Identity Center.

Démarrez un AWS session du portail d'accès

Avant d'exécuter une application qui y accède Services AWS, vous avez besoin d'une session de portail d' AWS accès active pour que le SDK utilise l'authentification IAM Identity Center pour résoudre les informations d'identification. En fonction de la durée de session que vous avez configurée, votre accès finira par expirer et le SDK rencontrera une erreur d'authentification. Pour vous connecter au portail AWS d'accès, exécutez la commande suivante dans le AWS CLI.

```
aws sso login
```

Si vous avez suivi les instructions et que vous avez configuré un profil par défaut, il n'est pas nécessaire d'appeler la commande avec une `--profile` option. Si la configuration de votre fournisseur de jetons SSO utilise un profil nommé, la commande est `aws sso login --profile named-profile`.

Pour éventuellement vérifier si vous avez déjà une session active, exécutez la AWS CLI commande suivante.

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

Si votre session est active, la réponse à cette commande indique le compte IAM Identity Center et l'ensemble d'autorisations configurés dans le `config` fichier partagé.

> **Note**
>
> Si vous disposez déjà d'une session active sur le portail AWS d'accès et que vous l'exécutez`aws sso login`, il ne vous sera pas demandé de fournir des informations d'identification.
> Le processus de connexion peut vous demander d'autoriser l' AWS CLI accès à vos données. Comme il AWS CLI repose sur le SDK pour Python (Boto), les messages d'autorisation peuvent contenir des variantes du `botocore` nom.

En savoir plus sur l'authentification

- Pour plus de détails sur l'utilisation d'IAM Identity Center pour l'authentification, voir Comprendre l'authentification IAM Identity Center dans le Guide de référence AWS des SDK et des outils
- Pour en savoir plus sur les bonnes pratiques, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.
- Pour créer des AWS informations d'identification à court terme, consultez la section Informations d'identification de sécurité temporaires dans le guide de l'utilisateur IAM.
- Pour en savoir plus sur les autres fournisseurs d'informations d'identification qui AWS SDK pour PHP peuvent les utiliser, consultez la section Fournisseurs d'informations d'identification standardisés dans le Guide de référence AWS des SDK et des outils.

Création d'une application simple à l'aide de la AWS SDK pour PHP version 3

Dites bonjour à Amazon S3 à l'aide du AWS SDK pour PHP. L'exemple suivant affiche la liste de vos compartiments Amazon S3.

Prérequis

- Téléchargez et installez le SDK
- Avant d'utiliser le AWS SDK pour PHP, vous devez configurer l'authentification avec AWS. Pour plus d'informations sur la configuration de l'authentification, voir Authentification avec AWS utilisant AWS SDK pour PHP Version 3

Inclure le SDK dans votre code

Quelle que soit la méthode que vous avez utilisée pour installer le kit SDK, vous pouvez inclure ce dernier dans votre code à l'aide d'une seule déclaration `require`. Consultez le tableau suivant pour savoir quel code PHP convient le mieux à votre installation technique. Remplacez toutes les occurrences de `/path/to/` par le chemin d'accès sur votre système.

Technique d'installation	Déclaration require
Composer	`require '/path/to/vendor/autoload.php';`
phar	`require '/path/to/aws.phar';`
ZIP	`require '/path/to/aws-autoloader.php';`

Dans cette rubrique, nous partons de la méthode d'installation de Composer. Si vous utilisez une autre méthode d'installation, vous pouvez revenir à cette section pour trouver le code `require` approprié à utiliser.

Ecriture du code

Assurez-vous de pouvoir vous authentifier.

Copiez et collez le code suivant dans un nouveau fichier source. Enregistrez le fichier et nommez-le`hello-s3.php`.

```
<?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";
}
```

Exécution du programme

Ouvrez une invite de commande pour exécuter votre programme PHP. La syntaxe de commande typique pour exécuter un programme PHP est la suivante :

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

Cet exemple de code n'utilise aucun argument. Pour exécuter ce code, entrez ce qui suit dans l'invite de commande :

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

Étapes suivantes

Pour tester de nombreuses autres opérations Amazon S3, consultez le référentiel d'exemples de AWS code sur GitHub.

À utiliser AWS Cloud9 avec la AWS SDK pour PHP version 3

> **Note**
>
> AWS Cloud9 n'est plus disponible pour les nouveaux clients. Les clients existants de AWS Cloud9 peuvent continuer à utiliser le service normalement. En savoir plus.

AWS Cloud9 est un environnement de développement intégré (IDE) basé sur le Web qui contient un ensemble d'outils que vous pouvez utiliser pour coder, créer, exécuter, tester, déboguer et publier

des logiciels dans le cloud. Vous pouvez utiliser AWS Cloud9 with AWS SDK pour PHP pour écrire et exécuter votre code PHP à l'aide d'un navigateur. AWS Cloud9 inclut des outils tels qu'un éditeur de code et un terminal. L' AWS Cloud9 IDE étant basé sur le cloud, vous pouvez travailler sur vos projets depuis votre bureau, votre domicile ou n'importe où en utilisant une machine connectée à Internet. Pour des informations générales à ce sujet AWS Cloud9, consultez le guide de AWS Cloud9 l'utilisateur.

Suivez ces instructions pour configurer AWS Cloud9 avec AWS SDK pour PHP :

- Étape 1 : Configurez votre appareil Compte AWS pour l'utiliser AWS Cloud9
- Étape 2 : Configuration de votre environnement AWS Cloud9 de développement
- Étape 3 : Configurez le AWS SDK pour PHP
- Étape 4 : Téléchargez un exemple de code
- Étape 5 : Exécuter un exemple de code

Étape 1 : Configurez votre appareil Compte AWS pour l'utiliser AWS Cloud9

Pour l'utiliser AWS Cloud9, connectez-vous à la AWS Cloud9 console depuis le Console de gestion AWS.

> **Note**
>
> Si vous utilisez AWS IAM Identity Center pour vous authentifier, vous devrez peut-être ajouter l'autorisation requise de à la politique attachée `iam:ListInstanceProfilesForRole` à l'utilisateur dans la console IAM.

Pour configurer une entité IAM dans votre AWS compte afin d'accéder à la AWS Cloud9 console AWS Cloud9 et de s'y connecter, consultez la section Configuration de l'équipe AWS Cloud9 dans le guide de l'AWS Cloud9 utilisateur.

Étape 2 : Configuration de votre environnement AWS Cloud9 de développement

Une fois connecté à la AWS Cloud9 console, utilisez-la pour créer un environnement de AWS Cloud9 développement. Après avoir créé l'environnement, AWS Cloud9 ouvre l'IDE correspondant à cet environnement.

Pour plus de détails, consultez la section Création d'un environnement AWS Cloud9 dans le guide de AWS Cloud9 l'utilisateur.

> **Note**
>
> Si vous créez votre environnement dans la console pour la première fois, nous vous recommandons de choisir l'option Create a new instance for environment (EC2) (Créer une nouvelle instance pour l'environnement (EC2)). Cette option indique AWS Cloud9 de créer un environnement, de lancer une instance Amazon EC2, puis de connecter la nouvelle instance au nouvel environnement. C'est le moyen le plus rapide de commencer à l'utiliser AWS Cloud9.

Si le terminal n'est pas déjà ouvert dans l'IDE, ouvrez-le. Choisissez Window, New Terminal (Fenêtre, Nouveau terminal) dans la barre de menus de l'IDE. Vous pouvez utiliser la fenêtre du terminal pour installer des outils et créer vos applications.

Étape 3 : Configurez le AWS SDK pour PHP

Après avoir AWS Cloud9 ouvert l'IDE pour votre environnement de développement, utilisez la fenêtre du terminal pour le configurer AWS SDK pour PHP dans votre environnement.

Composer est la méthode recommandée pour installer le AWS SDK pour PHP. Composer est un outil PHP qui gère et installe les dépendances de votre projet.

Pour en savoir plus sur l'installation de Composer, la configuration du chargement automatique et le respect d'autres bonnes pratiques pour définir des dépendances, rendez-vous sur getcomposer.org.

Installation de Composer

Si Composer n'est pas déjà dans votre projet, téléchargez et installez Composer sur la page Télécharger Composer.

- Pour Windows, suivez les instructions du programme d'installation de Windows.
- Pour Linux, suivez les instructions d'installation en ligne de commande.

Ajouter AWS SDK pour PHP en tant que dépendance via Composer

Si Composer est déjà installé globalement sur votre système, exécutez ce qui suit dans le répertoire de base de votre projet pour l'installer AWS SDK pour PHP en tant que dépendance :

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

Sinon, tapez cette commande Composer pour installer la dernière version du AWS SDK pour PHP en tant que dépendance.

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

Ajouter un chargeur automatique à vos scripts php

L'installation de Composer crée plusieurs dossiers et fichiers dans votre environnement. Le fichier principal que vous utiliserez est `autoload.php`, qui se trouve dans le dossier `vendor` de votre environnement.

Pour utiliser le AWS SDK pour PHP dans vos scripts, incluez le chargeur automatique dans vos scripts, comme suit.

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

Étape 4 : Téléchargez un exemple de code

Utilisez la fenêtre du terminal pour télécharger un exemple de code à utiliser AWS SDK pour PHP dans l'environnement de AWS Cloud9 développement.

Pour télécharger une copie de tous les exemples de code utilisés dans la documentation officielle du AWS SDK dans le répertoire racine de votre environnement, exécutez la commande suivante :

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

Les exemples de code pour le se AWS SDK pour PHP trouvent dans le `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` répertoire, où se `ENVIRONMENT_NAME` trouve le nom de votre environnement de développement.

Pour suivre l'exemple d'Amazon S3, nous vous recommandons de commencer par un exemple de code`ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php`. Cet exemple répertorie vos compartiments Amazon S3. Utilisez la fenêtre du terminal pour accéder au `s3` répertoire et répertorier les fichiers.

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

Pour ouvrir le fichier AWS Cloud9, vous pouvez cliquer sur le bouton `ListBuckets.php` directement dans la fenêtre du terminal.

Pour plus d'informations sur la compréhension des exemples de code, consultez la section Exemples de AWS SDK pour PHP code.

Étape 5 : Exécuter un exemple de code

Pour exécuter du code dans votre environnement de AWS Cloud9 développement, cliquez sur le bouton Exécuter dans la barre de menu supérieure. AWS Cloud9 détecte automatiquement l'extension du `.php` fichier et utilise le moteur d'exécution PHP (serveur Web intégré) pour exécuter le code. Cependant, pour cet exemple, nous voulons réellement l'option PHP (**`cli`**). Pour plus d'informations sur l'exécution de code dans AWS Cloud9, consultez la section Exécuter votre code dans le guide de AWS Cloud9 l'utilisateur.

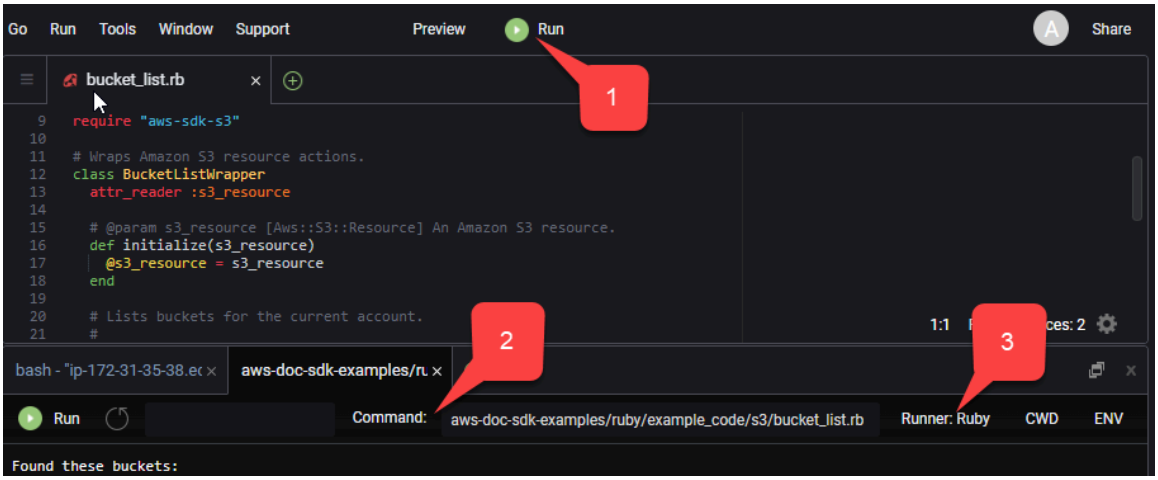
Dans la capture d'écran suivante, notez ces domaines de base :

- 1 : Courez. Le bouton Exécuter se trouve dans la barre de menu supérieure. Cela ouvre un nouvel onglet pour vos résultats.

  > **Note**
  >
  > Vous pouvez également créer manuellement de nouvelles configurations d'exécution. Dans la barre de menus, choisissez Exécuter, Configurations d'exécution, Nouvelle configuration d'exécution.

- 2 : Commande. AWS Cloud9 remplit la zone de texte Commande avec le chemin et le nom du fichier que vous exécutez. Si votre code s'attend à ce que des paramètres de ligne de commande soient transmis, ceux-ci peuvent être ajoutés à la ligne de commande de la même manière que vous le feriez lorsque vous exécutez le code via une fenêtre de terminal.

- 3 : Coureur. AWS Cloud9 détecte l'extension de votre fichier `.php` et sélectionne le PHP (serveur Web intégré) Runner pour exécuter votre code. Sélectionnez PHP (**`cli`**) pour exécuter cet exemple à la place.



Toute sortie générée à partir du code en cours d'exécution s'affiche dans l'onglet.

Configuration des clients de service dans la AWS SDK pour PHP version 3

Pour y accéder par programmation Services AWS, la AWS SDK pour PHP version 3 utilise un objet client pour chacune. Service AWS Par exemple, si votre application doit accéder à Amazon EC2, vous créez un objet EC2 client Amazon, une instance de la classe Ec2Client, pour interagir avec ce service. Vous utilisez ensuite le client du service pour y faire des demandes Service AWS.

Il existe de nombreuses manières de configurer le comportement du SDK, mais en fin de compte, tout dépend du comportement des clients du service. Toute configuration n'a aucun effet tant que votre code ne crée pas un client de service qui utilise la configuration.

Voici des exemples de configuration que vous fournissez :

- Comment votre code s'authentifie AWS lorsque vous appelez un service
- Le service que Région AWS vous souhaitez qu'un client utilise
- Paramètres de nouvelle tentative et de délai d'expiration pour les appels de service
- Configuration d'un proxy HTTP

Reportez-vous au guide de référence AWS SDKs et aux outils pour connaître les paramètres, les fonctionnalités et les autres concepts fondamentaux communs à de AWS SDKs nombreux.

Configuration externe des clients de service pour la AWS SDK pour PHP version 3

De nombreux paramètres de configuration peuvent être gérés en dehors de votre code. La plupart des paramètres de configuration peuvent être définis sous forme de variables d'environnement ou dans un AWS `config` fichier partagé distinct. Le `config` fichier AWS partagé peut gérer des ensembles de paramètres distincts, appelés profils, afin de fournir différentes configurations pour différents environnements ou tests. Pour une discussion complète sur le AWS partage `config` et les `credentials` fichiers, consultez le guide de référence des outils AWS SDKs et des outils.

La plupart des variables d'environnement et des paramètres de `config` fichiers partagés sont standardisés AWS SDKs et partagés entre des outils permettant de garantir des fonctionnalités cohérentes entre les différents langages de programmation et applications.

Pour voir tous les paramètres que le SDK peut résoudre à partir des variables d'environnement ou des fichiers de configuration, consultez la référence des paramètres dans le guide de référence des outils AWS SDKs et des outils.

Chaîne de fournisseurs de configuration pour la configuration du client

Le SDK vérifie plusieurs emplacements (ou sources) pour trouver des valeurs de configuration.

1. Tout paramètre explicite défini dans le code ou sur un client de service lui-même a priorité sur tout autre paramètre.
2. Variables d'environnement
   - Pour plus de détails sur la définition des variables d'environnement, voir les variables d'environnement dans le guide de référence AWS SDKs et Tools.
   - Notez que vous pouvez configurer des variables d'environnement pour un shell à différents niveaux de portée : à l'échelle du système, à l'échelle de l'utilisateur et pour une session de terminal spécifique.
3. Partage `config` et `credentials` fichiers
   - Pour plus de détails sur la configuration de ces fichiers, consultez les sections Shared `config` et `credentials` files du guide de référence AWS SDKs and Tools.
4. Toute valeur par défaut fournie par le code source du SDK lui-même est utilisée en dernier.
   - Certaines propriétés, telles que Region, n'ont pas de valeur par défaut. Vous devez les spécifier de manière explicite dans le code, dans un paramètre d'environnement ou dans le `config` fichier partagé. Si le SDK ne parvient pas à résoudre la configuration requise, les demandes d'API peuvent échouer lors de l'exécution.

Outre cette chaîne de configuration générale, la AWS SDK pour PHP version 3 utilise également des chaînes de fournisseurs spécialisées, notamment la chaîne de fournisseurs d'informations d'identification et la chaîne de Région AWS résolution. Ces chaînes spécialisées ajoutent des fournisseurs supplémentaires qui tiennent compte de l'environnement dans lequel le SDK s'exécute. Par exemple, dans un conteneur ou une instance EC2.

Création d'un client de service configuré à l'aide de paramètres externes

Vous devez créer un client de service dans votre application pour parler à un Service AWS. Les clients du service sont votre lien essentiel. Services AWS Ils gèrent tous les détails complexes des communications afin que vous n'ayez pas à vous soucier d'eux. Ils prennent en charge des tâches

importantes telles que la sécurité, la gestion des erreurs et les nouvelles tentatives automatiquement, ce qui vous permet de vous concentrer sur le développement de votre application plutôt que sur les complications techniques.

Construire un client de service à l'aide d'un constructeur sans paramètres

Si tous les paramètres de configuration dont vous avez besoin proviennent de sources externes, vous pouvez créer un client de service en utilisant un constructeur vide :

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

use Aws\S3\S3Client;

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

L'extrait de code précédent crée une `S3Client` instance. Lors de la création, le SDK analyse les paramètres dans la chaîne des fournisseurs de configuration. Une fois que le SDK trouve une valeur de paramètre, il l'utilise.

La chaîne Région AWS de fournisseurs par défaut et la chaîne de fournisseurs d'informations d'identification par défaut seront également utilisées dans le processus de création. À un stade quelconque de la chaîne, le SDK doit résoudre les paramètres Région AWS à utiliser et à rechercher qui lui permettent de récupérer les informations d'identification pour les demandes de signature. Si le SDK ne trouve pas ces valeurs, la création du client échoue.

AWS SDK pour PHP Variables d'environnement version 3

Au-delà des paramètres cross-sdk pris en charge par la plupart AWS SDKs, la AWS SDK pour PHP version 3 fonctionne avec les variables d'environnement suivantes :

AWS_SDK_CHARGER_NONDEFAULT_CONFIG

: Lorsqu'elle est définie, cette variable d'environnement indique au SDK de charger les informations d'identification à partir du `~/.aws/credential` ou des AWS `config` fichiers en plus du ou des `credentials` fichiers. `~/.aws/config`

AWS_SDK_UA_APP_ID

: Définit un identifiant d'application personnalisé qui est inclus dans l'en-tête User-Agent pour les demandes effectuées par le SDK.

AWS_SUPPRESS_AVERTISSEMENT D'OBSOLESCENCE PHP

Lorsqu'il est défini sur true, il supprime les avertissements d'obsolescence PHP que le SDK pourrait générer.

Configuration des clients de service dans le code pour la AWS SDK pour PHP version 3

Au lieu ou en plus de la configuration externe des clients de service, vous pouvez les configurer par programmation dans le code.

En configurant les clients de service dans le code, vous contrôlez avec précision les nombreuses options qui s'offrent à vous. La plupart des configurations que vous pouvez définir en externe peuvent également être définies dans le code.

Configuration de base dans le code

Vous pouvez créer et configurer un client de service dans le code en transmettant un tableau associatif d'options au constructeur d'un client. Dans l'exemple suivant, le tableau associatif contient uniquement l'option « region » utilisée par le client :

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

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

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

Les informations relatives au paramètre optionnel « version » sont disponibles dans la rubrique des options de configuration.

Notez que nous n’avons pas explicitement fourni d’informations d’identification au client. Cela est dû au fait que le SDK utilise la chaîne de fournisseurs d'informations d'identification par défaut pour rechercher les informations d'identification.

Toutes les options générales de configuration du client sont décrites en détail dansOptions du constructeur client pour la AWS SDK pour PHP version 3. Les options fournies peuvent varier selon

le type de client que vous créez. Ces options de configuration client personnalisées sont décrites dans la documentation sur l'API de chaque client.

Utilisation de la **Sdk** classe

La classe `Aws\Sdk` fonctionne comme une fabrique de clients. Elle permet de gérer des options de configuration partagées sur plusieurs clients. La plupart des options qui peuvent être fournies à un constructeur client spécifique peuvent également être fournies à la `Aws\Sdk` classe. Ces options sont ensuite appliquées à chaque constructeur client.

Importations

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

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

Exemple de code

```
// 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();
```

Les options partagées entre tous les clients sont placées dans des paires clé-valeur au niveau racine. Les données de configuration spécifiques au service peuvent être fournies dans un tableau associatif avec une clé identique à l'espace de noms d'un service (par exemple, « S3 », « », etc.)DynamoDb.

```
$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();
```

Les valeurs de configuration spécifiques au service comprennent les valeurs spécifiques au service et les valeurs de niveau racine (les valeurs spécifiques au service sont fusionnées de manière superficielle dans les valeurs de niveau racine).

> **Note**
>
> Nous vous recommandons vivement d'utiliser la classe `Sdk` pour créer des clients si vous utilisez plusieurs instances client dans votre application. La classe `Sdk` utilise automatiquement le même client HTTP pour chaque client de kit SDK. Ainsi, des clients de kit SDK pour différents services peuvent lancer des requêtes HTTP non bloquantes. Si les clients de kit SDK n'utilisent pas le même client HTTP, les demandes HTTP envoyées par le client de kit SDK peuvent bloquer l'orchestration des promesses entre les services.

Options du constructeur client pour la AWS SDK pour PHP version 3

Les options du constructeur client peuvent être fournies dans un constructeur client ou fournies à la `Aws\Sdk`classe. Les options fournies peuvent varier selon le type de client que vous créez. Ces options de configuration client personnalisées sont décrites dans la documentation sur l'API de chaque client.

Si vous ne fournissez pas explicitement les options de constructeur client dont le client a besoin, le SDK for PHP recherche les valeurs des variables d'environnement ou d' AWS un fichier de configuration. Tous les clients ont besoin d'une valeur de fournisseur d'informations d'identification et d'une Région AWS valeur. Vous devez donc définir ces valeurs en tant qu'options du constructeur ou en externe.

Par défaut, le fichier de configuration en cours de vérification est `.aws/config` dans votre répertoire personnel, généralement `~/.aws/config`. Toutefois, vous pouvez utiliser la variable d'environnement `AWS_CONFIG_FILE` pour définir l'emplacement de votre fichier de configuration par défaut. Par exemple, cela peut être utile si vous limitez l'accès aux fichiers à certains répertoires avec`open_basedir`.

Pour plus d'informations sur l'emplacement et le formatage des `credentials` fichiers partagés AWS `config`, consultez la section Configuration dans le guide de référence des outils AWS SDKs et.

Pour plus de détails sur tous les paramètres de configuration globaux que vous pouvez définir dans les fichiers de AWS configuration ou en tant que variables d'environnement, consultez la section Référence des paramètres de configuration et d'authentification dans le guide de référence des outils AWS SDKs et.

Options de configuration

- api_provider
- informations d’identification
- debug
- stats
- point de terminaison
- endpoint_provider
- endpoint_discovery
- handler
- http
- http_handler
- profile
- region
- nouvelles tentatives
- scheme
- web
- signature_provider
- signature_version
- ua_append
- utilisez_aws_shared_config_files
- valider
- version

L'exemple suivant montre comment transmettre des options au constructeur d'un client Amazon S3.

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

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

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

Consultez le the section called “Création d'un client de service de base” pour plus d'informations sur la création de clients.

api_provider

Type

`callable`

Fonction PHP de type callable qui accepte un type, un service et un argument de version, et renvoie un tableau de données de configuration correspondantes. Son type peut prendre l'une des valeurs suivantes : `api`, `waiter` ou `paginator`.

Par défaut, le kit SDK utilise une instance de `Aws\Api\FileSystemApiProvider` qui charge des fichiers API depuis le dossier `src/data` du kit SDK.

informations d’identification

Type

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

Spécifiez un objet `Aws\Credentials\CredentialsInterface` pour utiliser une instance d'informations d'identification spécifique. Ce qui suit indique que le fournisseur d'informations d'identification IAM Identity Center doit être utilisé. Ce fournisseur est également connu sous le nom de fournisseur d'informations d'identification SSO.

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

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

```
    'credentials' => $credentials
]);
```

Si vous utilisez un profil nommé, remplacez le nom de votre profil par « `default` » dans l'exemple précédent. Pour en savoir plus sur la configuration de profils nommés, consultez la section Shared `config` and `credentials` files dans le guide de référence AWS SDKs and Tools.

Si vous ne spécifiez pas de fournisseur d'informations d'identification à utiliser et que vous vous fiez à la chaîne de fournisseurs d'informations d'identification, le message d'erreur résultant de l'échec de l'authentification est généralement générique. Il est généré à partir du dernier fournisseur de la liste des sources dont les informations d'identification sont valides, ce qui n'est peut-être pas le fournisseur que vous essayez d'utiliser. Lorsque vous spécifiez le fournisseur d'informations d'identification à utiliser, le message d'erreur qui en résulte est plus utile et pertinent car il provient uniquement de ce fournisseur. Pour en savoir plus sur la chaîne de sources dont les informations d'identification sont vérifiées, consultez la section Chaîne de fournisseurs d'informations d'identification dans le guide de référence AWS SDKs et Tools.

Spécifiez `false` pour utiliser des informations d'identification null et si vous ne souhaitez pas signer les requêtes.

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

Spécifiez une fonction de type callable fournisseur d’informations d’identification pour créer des informations d’identification à l’aide d’une fonction.

```
use Aws\Credentials\CredentialProvider;

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

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

Spécifiez une instance de `Aws\CacheInterface` pour mettre en cache les valeurs renvoyées par la chaîne du fournisseur par défaut dans plusieurs processus.

```
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
]);
```

Vous trouverez plus d'informations sur la fourniture d'informations d'identification à un client dans le guide Credentials for the AWS SDK pour PHP Version 3.

> **Note**
>
> Les informations d'identification sont chargées et validées lentement lorsqu'elles sont utilisées.

debug

Type

`bool|array`

Génère des informations de débogage sur chaque transfert. Les informations de débogage contiennent des informations sur chaque changement d'état d'une transaction lors de sa préparation et de son envoi sur le réseau. La sortie de débogage comprend également des informations sur le gestionnaire HTTP spécifique utilisé par un client (par exemple, sortie de débogage cURL).

Définissez cette option sur `true` pour afficher des informations de débogage lors de l'envoi de requêtes.

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

```
]);

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

Vous pouvez également fournir un tableau associatif contenant les clés suivantes.

logfn (callable)

Fonction appelée avec des messages de journaux. La fonction PHP `echo` est utilisée par défaut.

stream_size (int)

Lorsque la taille d'un flux est supérieure à ce nombre, les données du flux ne sont pas consignées. Définissez cette option sur `0` si vous ne souhaitez pas consigner les données du flux.

scrub_auth (bool)

Réglez sur `false` pour désactiver le nettoyage des données d'authentification contenues dans les messages enregistrés (cela signifie que l'identifiant et la signature de votre clé d' AWS accès seront transmis au`logfn`).

http (bool)

Définissez cette option sur `false` pour désactiver la fonction de « débogage (debug) » des gestionnaires HTTP de niveau inférieur (par exemple, sortie cURL verbose).

auth_headers (array)

Définissez un mappage clé-valeur des en-têtes que vous souhaitez remplacer avec la valeur par laquelle vous souhaitez les remplacer. Ces valeurs ne sont pas utilisées, sauf si `scrub_auth` est défini sur `true`.

auth_strings (array)

Définissez un mappage clé-valeur des expressions régulières à mapper avec leurs remplacements. Ces valeurs sont utilisées par le nettoyeur de données d'authentification si `scrub_auth` est défini sur `true`.

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => [
        'logfn'        => function ($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**
>
> Cette option produit également les informations du gestionnaire HTTP sous-jacent produites par l'option de `http` débogage. La sortie de débogage s'avère très utile pour diagnostiquer les problèmes dans le kit AWS SDK pour PHP. Veuillez fournir la sortie de débogage d'un échec isolé lorsque vous ouvrez une question concernant le kit SDK.

stats

Type

`bool|array`

Associe les statistiques de transfert aux erreurs et aux résultats renvoyés par les opérations du kit SDK.

Définissez cette option sur `true` pour collecter des statistiques de transfert sur les requêtes envoyées.

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

// Perform an operation
$result = $s3->listBuckets();
```

```
// Inspect the stats
$stats = $result['@metadata']['transferStats'];
```

Vous pouvez également fournir un tableau associatif contenant les clés suivantes.

retries (bool)

Définissez ce paramètre sur `false` pour désactiver le signalement des tentatives de tentative. Les statistiques sur les nouvelles tentatives sont collectées et renvoyées par défaut.

http (bool)

Définissez sur pour `true` permettre la collecte de statistiques à partir d'adaptateurs HTTP de niveau inférieur (par exemple, les valeurs renvoyées GuzzleHttpTransferStats). Les gestionnaires HTTP doivent prendre en charge une option __on_transfer_stats pour que cette option fonctionne. Les statistiques HTTP sont renvoyées sous la forme d'un ensemble indexé de tableaux associatifs. Chaque tableau contient les statistiques de transfert renvoyées pour une requête par le gestionnaire HTTP du client. Ce paramètre est désactivé par défaut.

Si une nouvelle tentative a été effectuée pour une demande, toutes les statistiques de transfert sont renvoyées. `$result['@metadata']['transferStats']['http'][0]` contient alors les statistiques de la première demande, `$result['@metadata']['transferStats']['http'][1]` contient celles de la deuxième demande, etc.

timer (bool)

Définissez cette option sur `true` pour activer un minuteur de commande qui indique le temps total passé sur une opération (en secondes). Ce paramètre est désactivé par défaut.

```
$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'];
```

point de terminaison

Type

`string`

URI complet du service web. Cela est nécessaire pour les services, tels que ceux qui utilisent des AWS Elemental MediaConvertpoints de terminaison spécifiques au compte. Pour ces services, demandez ce point de terminaison à l'aide de la `describeEndpoints` méthode.

Cela n'est nécessaire que lors de la connexion à un point de terminaison personnalisé (par exemple, une version locale d'Amazon S3 ou Amazon DynamoDB Local).

Voici un exemple de connexion à Amazon DynamoDB Local :

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

Consultez les AWS régions et points de terminaison pour obtenir la liste des AWS régions et points de terminaison disponibles.

endpoint_provider

Type

`Aws\EndpointV2\EndpointProviderV2|callable`

Une instance optionnelle de EndpointProvider V2 ou PHP appelable qui accepte un hachage d'options, y compris une clé « service » et une clé « région ». Cet élément renvoie `NULL` ou un hachage de données de point de terminaison, pour lequel la clé « endpoint » est obligatoire.

Voici un exemple de création d'un fournisseur de point de terminaison minimal.

```
$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

Type

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

La découverte de points de terminaison identifie et se connecte au point de terminaison correct pour une API de service qui prend en charge la découverte de points de terminaison. Pour les services qui prennent en charge mais ne nécessitent pas la découverte de points de terminaison, activez `endpoint_discovery` lors de la création du client. Si un service ne prend pas en charge la découverte de points de terminaison, cette configuration est ignorée.

`Aws\EndpointDiscovery\ConfigurationInterface`

Un fournisseur de configuration facultatif qui permet la connexion automatique au point de terminaison approprié d'une API de service pour les opérations que le service spécifie.

L'objet `Aws\EndpointDiscovery\Configuration` accepte deux options, dont une valeur booléenne, « enabled », qui indique si la découverte de points de terminaison est activée, et un nombre entier « cache_limit », qui indique le nombre maximal de clés dans le cache des points de terminaison.

Pour chaque client créé, transmettez un objet `Aws\EndpointDiscovery\Configuration` pour utiliser une configuration spécifique pour la découverte de points de terminaison.

```
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,

]);
```

Spécifiez une instance de `Aws\CacheInterface` pour mettre en cache les valeurs renvoyées par la découverte de points de terminaison dans plusieurs processus.

```
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),
]);
```

Transmettez un tableau à la découverte de points de terminaison.

```
use Aws\S3\S3Client;

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

handler

Type

`callable`

Gestionnaire qui accepte un objet de commande et un objet de requête, et qui renvoie une promesse (`GuzzleHttp\Promise\PromiseInterface`) exécutée avec un objet `Aws\ResultInterface` ou rejetée avec `Aws\Exception\AwsException`. Un gestionnaire n'accepte pas un gestionnaire suivant, car il s'agit d'un élément terminal qui doit traiter une commande. Si aucun gestionnaire n'est fourni, un gestionnaire Guzzle par défaut est utilisé.

`Aws\MockHandler` vous permet de renvoyer des résultats fictifs ou de lancer des exceptions de simulation. Vous mettez en file d'attente les résultats ou les exceptions, puis MockHandler vous les retirerez dans l'ordre FIFO.

```
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

Type

```
array
```

Défini sur un tableau d'options HTTP appliquées aux requêtes et transferts HTTP créés par le kit SDK.

Le kit SDK prend en charge les options de configuration suivantes :

cert

Type

```
string|array
```

Spécifier le certificat au format PEM côté client.

- Définir en tant que chaîne pour le chemin d'accès au fichier de certificat uniquement.

```
use Aws\S3\S3Client;

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

- Définir en tant que tableau contenant le chemin d'accès et mot de passe.

```
use Aws\S3\S3Client;

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

connect_timeout

Valeur de type float indiquant le temps d'attente en secondes lors d'une tentative de connexion à un serveur. Spécifiez la valeur `0` pour indiquer une durée indéfinie (comportement par défaut).

```
use Aws\DynamoDb\DynamoDbClient;

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

debug

Type

`bool|resource`

Indique au gestionnaire HTTP sous-jacent de générer des informations de débogage. Les informations de débogage fournies varient selon les gestionnaires HTTP.

- Spécifiez `true` pour écrire la sortie de débogage dans STDOUT.
- Spécifiez `resource`, comme renvoyé par `fopen`, pour écrire la sortie de débogage dans une ressource de flux PHP spécifique.

decode_content

Type

`bool`

Indique au gestionnaire HTTP sous-jacent de gonfler le corps des réponses compressées. Lorsque cette option est désactivée, les corps des réponses compressées peuvent être gonflés à l'aide de `GuzzleHttp\Psr7\InflateStream`.

Note

Le décodage du contenu est activé par défaut dans le gestionnaire HTTP par défaut du kit SDK. Pour des raisons de rétrocompatibilité, ce paramétrage ne peut pas être modifié. Si vous stockez des fichiers compressés dans Amazon S3, nous vous recommandons de désactiver le décodage du contenu au niveau du client S3.

```
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

Type

`int`

Durée d'attente avant l'envoi de la requête (en millisecondes). Cette option est souvent utilisée pour retarder une nouvelle tentative de requête.

expect

Type

`bool|string`

Cette option est transmise au gestionnaire HTTP sous-jacent. Par défaut, l'en-tête Expect: 100-Continue est défini lorsque le corps de la demande dépasse 1 Mo. `true` ou `false` active ou désactive l'en-tête sur toutes les demandes. Si un nombre entier est utilisé, seules les demandes avec des corps qui dépassent ce paramètre utiliseront l'en-tête. Lorsqu'elle est utilisée en tant que nombre entier, si la taille du corps est inconnue, l'en-tête Expect sera envoyé.

> ⚠ Warning
>
> La désactivation de l'en-tête Expect peut empêcher le service de renvoyer l'authentification ou d'autres erreurs. Cette option doit être configurée avec précautions.

progress

Type

`callable`

Définit une fonction à appeler en cas d'avancement du transfert. La fonction accepte les arguments suivants :

1. Nombre total d'octets à télécharger estimé.
2. Nombre d'octets téléchargés jusqu'à présent.
3. Nombre d'octets à charger estimé.
4. Nombre d'octets chargés jusqu'à présent.

```
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

Type

`string|array`

Vous pouvez vous connecter à un AWS service via un proxy en utilisant l'`proxy`option.

- Fournissez une valeur de chaîne pour vous connecter à un proxy pour tous les types de URIs. La valeur de type chaîne proxy peut contenir un schéma, un nom d'utilisateur et un mot de passe. Par exemple, "`http://username:password@192.168.16.1:10`".
- Spécifiez un tableau associatif de paramètres proxy dans lequel la clé correspond au schéma de l'URI et la valeur correspond au proxy de l'URI donné (vous pouvez ainsi accorder différents proxys pour les points de terminaison « http » et « https »).

```
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',
        ]
    ]
]);
```

Vous pouvez utiliser la variable d'environnement `HTTP_PROXY` pour configurer un proxy spécifique au protocole « http » et la variable d'environnement `HTTPS_PROXY` pour configurer un proxy spécifique au protocole « https ».

sink

Type

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

L'option `sink` permet de contrôler l'emplacement dans lequel les données de réponse d'une opération sont téléchargées.

- Spécifiez `resource`, comme renvoyé par `fopen`, pour télécharger le corps de la réponse dans un flux PHP.
- Spécifiez le chemin d'accès d'un fichier sur disque sous la forme d'une valeur `string` pour télécharger le corps de la réponse dans ce fichier.
- Spécifiez `Psr\Http\Message\StreamInterface` pour télécharger le corps de la réponse dans un objet de flux PSR spécifique.

> **Note**
>
> Le kit SDK télécharge le corps de la réponse dans un flux temporaire PHP par défaut. Ainsi, les données restent en mémoire jusqu'à ce que la taille du corps atteigne 2 Mo, puis elles sont écrites dans un fichier temporaire sur disque.

synchronous

Type

`bool`

L'option `synchronous` informe le gestionnaire HTTP sous-jacent que vous souhaitez bloquer le résultat.

stream

Type

`bool`

Définissez cette option sur `true` pour indiquer au gestionnaire HTTP sous-jacent que vous souhaitez diffuser le corps d'une réponse depuis le service web, plutôt que de tout télécharger immédiatement. Par exemple, cette option est utilisée dans la classe Streamwrapper Amazon S3 pour garantir la diffusion des données.

timeout

Type

`float`

Valeur de type float indiquant le délai d'expiration de la requête (en secondes). Spécifiez la valeur `0` pour indiquer une durée indéfinie (comportement par défaut).

```
use Aws\DynamoDb\DynamoDbClient;

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

vérifier

Type

`bool|string`

Vous pouvez personnaliser le comportement de vérification des SSL/TLS certificats homologues du SDK à l'aide de `verify` `http` cette option.

- Définissez sur `true` pour activer la vérification des certificats SSL/TLS homologues et utiliser le bundle CA par défaut fourni par le système d'exploitation.
- Définissez cette option sur `false` pour désactiver la vérification des certificats pairs. (Cette méthode n'est pas sécurisée !)
- Définissez cette option sur une chaîne pour indiquer le chemin d'accès à un bundle d'autorité de certification (CA) afin d'activer la vérification à l'aide de ce bundle personnalisé.

Si le bundle d'autorité de certification (CA) est introuvable pour votre système et si vous recevez une erreur, indiquez le chemin d'accès à un bundle d'autorité de certification pour le kit SDK. Si vous n'avez pas besoin d'un bundle d'autorité de certification spécifique (CA), Mozilla en propose un couramment utilisé, que vous pouvez télécharger ici (géré par le mainteneur de cURL). Une fois que vous disposez d'un bundle d'autorité de certification (CA) sur un disque, vous pouvez définir le paramètre .ini PHP `openssl.cafile` de sorte qu'il pointe vers le chemin d'accès du fichier, ce qui vous permet d'ignorer l'option `verify` pour la requête. Vous trouverez plus de détails sur les certificats SSL sur le site web de cURL.

```
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

Type

```
callable
```

L'option `http_handler` permet d'intégrer le kit SDK à d'autres clients HTTP. Une option `http_handler` est une fonction qui accepte un objet `Psr\Http\Message\RequestInterface` et un ensemble d'options `http` appliquées à la commande. Elle renvoie un objet `GuzzleHttp\Promise\PromiseInterface` qui est exécuté avec un objet `Psr\Http\Message\ResponseInterface` ou rejeté avec un ensemble des données d'exception suivantes :

- `exception` - (`\Exception`) : exception rencontrée.
- `response` - (`Psr\Http\Message\ResponseInterface`) : réponse reçue (le cas échéant).
- `connection_error` - (bool) défini sur `true` pour marquer l'erreur comme une erreur de connexion. Si vous définissez cette valeur sur `true`, le kit SDK retente automatiquement l'opération, si nécessaire.

Le kit SDK convertit automatiquement l'option `http_handler` donnée en option `handler` normale en encapsulant l'option `http_handler` fournie avec un objet `Aws\WrappedHttpHandler`.

Par défaut, le kit SDK utilise Guzzle comme gestionnaire HTTP. Vous pouvez fournir un gestionnaire HTTP différent ici, ou fournir à un client Guzzle vos propres options personnalisées.

Définition de la version de TLS

Un cas d'utilisation consiste à définir la version de TLS utilisée par Guzzle avec Curl, en supposant que Curl est installé dans votre environnement. Notez les contraintes de version Curl concernant les versions de TLS prise en charge. Par défaut, c'est la dernière version qui est utilisée. Si la version de TLS est explicitement définie et que le serveur distant ne prend pas en charge cette version, il génère une erreur au lieu d'utiliser une version de TLS antérieure.

Vous pouvez déterminer la version de TLS utilisée pour une opération client donnée en définissant l'option client `debug` sur true et en examinant la sortie de connexion SSL. Cette ligne peut ressembler à ce qui suit : `SSL connection using TLSv1.2`

Exemple de définition de TLS 1.2 avec Guzzle 6 :

```
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**
>
> L'option `http_handler` prévaut sur toutes les autres options `handler` fournies.

profile

Type

`string`

L'option « profile » indique le profil à utiliser lorsque les informations d'identification sont créées à partir du fichier AWS d'informations d'identification de votre répertoire HOME (généralement`~/.aws/credentials`). Ce paramètre remplace la variable d'environnement `AWS_PROFILE`.

> **Note**
>
> Lorsque vous spécifiez l'option « profile », celle-ci est ignorée et les `"credentials"` paramètres relatifs aux informations d'identification dans le fichier de AWS configuration sont (généralement`~/.aws/config`) ignorés.

```
// Use the "production" profile from your credentials file
$ec2 = new Aws\Ec2\Ec2Client([
```

```
    'version' => '2014-10-01',
    'region'  => 'us-west-2',
    'profile' => 'production'
]);
```

Voir Informations d'identification pour la AWS SDK pour PHP version 3 pour plus d'informations sur la configuration des informations d'identification et le format de fichier .ini.

region

Type

`string`

Obligatoire

true

AWS Région à laquelle se connecter. Consultez les AWS régions et points de terminaison pour obtenir la liste des régions disponibles.

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

nouvelles tentatives

Type

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

Par défaut

`int(3)`

Configure le mode de nouvelles tentatives et le nombre maximal de tentatives autorisées pour un client. Spécifiez `0` pour désactiver les nouvelles tentatives.

Les trois modes de nouvelle tentative sont les suivants :

- `legacy`- l'ancienne implémentation par défaut des nouvelles tentatives
- `standard`- ajoute un système de quota de nouvelles tentatives pour empêcher les nouvelles tentatives qui ont peu de chances de réussir
- `adaptive` - s'appuie sur le mode standard, en ajoutant un limiteur de débit côté client. Notez que ce mode est considéré comme expérimental.

La configuration des tentatives est composée du mode et du nombre maximal de tentatives à utiliser pour chaque demande. La configuration peut être définie dans plusieurs emplacements différents, dans l'ordre de priorité suivant.

Ordre de priorité

L'ordre de priorité pour la nouvelle configuration est le suivant (1 remplace 2-3, etc.) :

1. Option de configuration du client
2. Variables d’environnement
3. AWS Fichier de configuration partagé

Variables d'environnement

- `AWS_RETRY_MODE` - défini sur `legacy`, `standard` ou `adaptive`
- `AWS_MAX_ATTEMPTS` - défini sur une valeur entière pour le maximum de tentatives par demande

Clés du fichier de configuration partagé

- `retry_mode` - défini sur `legacy`, `standard` ou `adaptive`
- `max_attempts` - défini sur une valeur entière pour le maximum de tentatives par demande

Configuration du client

L'exemple suivant désactive les nouvelles tentatives pour le client Amazon DynamoDB.

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

```
]);
```

L'exemple suivant passe dans un entier, qui sera par défaut sur le mode `legacy` avec le nombre de tentatives passé

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

L'objet **`Aws\Retry\Configuration`** accepte deux paramètres, le mode de nouvelles tentatives

et un entier pour le maximum de tentatives par demande. Cet exemple passe dans un

objet `Aws\Retry\Configuration` pour la configuration des nouvelles tentatives.

```
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,
]);
```

Cet exemple passe dans un tableau pour la configuration des nouvelles tentatives.

```
use Aws\S3\S3Client;

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

```
]);
```

Cet exemple passe une instance de `Aws\CacheInterface` pour mettre en cache les valeurs renvoyées par le fournisseur de configuration des nouvelles tentatives par défaut.

```
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

Type

`string`

Par défaut

`string(5) "https"`

Schéma d'URI à utiliser lors de la connexion. Le kit SDK utilise des points de terminaison « https » (connexions SSL/TLS) par défaut. Vous pouvez essayer de vous connecter à un service via un point de terminaison « http » non chiffré en définissant `scheme` sur « http ».

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

Consultez les AWS régions et les points de terminaison pour obtenir la liste des points de terminaison et savoir si un service prend en charge le `http` schéma.

web

Type

`string`

Obligatoire

true

Nom du service à utiliser. Cette valeur est renseignée par défaut lorsque vous utilisez un client fourni par le kit SDK (`Aws\S3\S3Client`). Cette option s'avère utile lorsque vous testez un service qui n'a pas encore été publié dans le kit SDK, mais dont vous disposez sur un disque.

signature_provider

Type

`callable`

Un appelable qui accepte un nom de version de signature (par exemple,`v4`), un nom de service et une AWS région et qui renvoie un `Aws\Signature\SignatureInterface` objet ou `NULL` si le fournisseur est en mesure de créer un signataire pour les paramètres donnés. Ce fournisseur permet de créer des utilisateurs utilisés par le client.

Le kit SDK propose plusieurs fonctions dans la classe `Aws\Signature\SignatureProvider` qui permettent de créer des fournisseurs de signatures personnalisés.

signature_version

Type

`string`

Chaîne représentant une version de signature personnalisée à utiliser avec un service (par exemple, `v4`, etc.). Une version de signature par opération PEUT remplacer cette version de signature demandée, si nécessaire.

Les exemples suivants montrent comment configurer un client Amazon S3 pour utiliser la version 4 de signature :

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

```
    'signature_version' => 'v4'
]);
```

> **Note**
>
> L'option `signature_provider` utilisée par votre client DOIT pouvoir créer l'option `signature_version` que vous spécifiez. L'option `signature_provider` utilisée par défaut par le kit SDK peut créer des objets de signature pour les versions de signature « anonymous » et « v4 ».

ua_append

Type

`string|string[]`

Par défaut

`[]`

Chaîne ou tableau de chaînes ajoutées à la chaîne de l'agent utilisateur transmise au gestionnaire HTTP.

utilisez_aws_shared_config_files

Type

`bool|array`

Par défaut

`bool(true)`

Définissez cette valeur sur false pour désactiver la vérification du fichier de configuration partagé dans « ~/ ». aws/config' and '~/.aws/credentials'. Cela remplacera la variable d' AWS_CONFIG_FILE environnement.

valider

Type

`bool|array`

Par défaut

`bool(true)`

Définissez cette option sur `false` pour désactiver la validation des paramètres côté client. Vous pouvez constater une légère amélioration au niveau des performances du client lorsque la validation est désactivée, mais la différence est négligeable.

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

Définissez cette option sur un tableau associatif d'options de validation pour activer des contraintes de validation spécifiques :

- `required` - Vérifie que les paramètres obligatoires sont présents (activé par défaut).
- `min` - Vérifie la longueur minimale d'une valeur (activé par défaut).
- `max` - Vérifie la longueur maximale d'une valeur.
- `pattern` - Vérifie que la valeur correspond à une expression régulière.

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

version

Type

`string`

Obligatoire

false

Cette option indique la version du service Web à utiliser (par exemple,`2006-03-01`).

À partir de la version 3.277.10 du SDK, l'option « version » n'est pas requise. Si vous ne spécifiez pas l'option « version », le SDK utilise la dernière version du client de service.

Deux situations nécessitent un paramètre « version » lorsque vous créez un client de service.

- Vous utilisez une version du SDK PHP antérieure à la version 3.277.10.
- Vous utilisez la version 3.277.10 ou ultérieure et souhaitez utiliser une version autre que la « dernière » pour un client de service.

Par exemple, l'extrait suivant utilise la version 3.279.7 du SDK, mais pas la dernière version du. `Ec2Client`

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

La spécification d'une contrainte de version vous permet de vous assurer que les modifications avec rupture apportées au service n'affecteront pas votre code.

Vous pouvez consulter la liste des versions d'API disponibles sur la page de documentation de l'API de chaque client. Si vous ne parvenez pas à charger une version d'API spécifique, il se peut que vous deviez mettre à jour votre copie du kit SDK.

Configuration du Région AWS pour la AWS SDK pour PHP version 3

Les clients du SDK se connectent à un Service AWS in spécifique Région AWS que vous spécifiez lors de la création du client. Cette configuration permet à votre application d'interagir avec AWS les ressources de cette zone géographique. Lorsque vous créez un client de service sans définir explicitement de région, le SDK utilise la région par défaut de votre configuration externe.

Chaîne de résolution régionale

La AWS SDK pour PHP version 3 utilise l'ordre suivant pour déterminer la région utilisée par un client de service :

1. Région fournie dans le code : si vous définissez explicitement la région dans les options du constructeur du client, celle-ci a priorité sur toutes les autres sources.

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

2. Variables d'environnement : si aucune région n'est fournie dans le code, le SDK vérifie la présence de ces variables d'environnement dans l'ordre suivant :
   - `AWS_REGION`
   - `AWS_DEFAULT_REGION`

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

3. AWS fichiers de configuration : si aucune variable d'environnement de région n'est définie, le SDK vérifie les AWS fichiers de configuration :
   a. Le SDK recherche `~/.aws/config` (ou l'emplacement spécifié par la variable d'`AWS_CONFIG_FILE`environnement)
   b. Le SDK examine le paramètre de région dans le profil spécifié par la variable d'`AWS_PROFILE`environnement
   c. Si non `AWS_PROFILE` est spécifié, le SDK utilise le profil « par défaut »

   Par exemple, supposons que nous disposions des paramètres de fichier de configuration suivants :

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

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

Si la variable d'`AWS_PROFILE`environnement est définie avec la valeur « production », les clients utilisent le`eu-central-1 Region`. S'il n'existe aucune variable d'`AWS_PROFILE`environnement, les clients utilisent la `eu-west-1` région.

4. Si le SDK ne trouve aucune valeur de région dans l'une des sources ci-dessus, il lance une exception car une valeur de région est un paramètre obligatoire pour un client de service.

Bonnes pratiques

Tenez compte des meilleures pratiques suivantes lorsque vous travaillez avec des régions dans la AWS SDK pour PHP version 3 :

Définissez explicitement la région dans le code de production

Pour les applications de production, nous vous recommandons de définir explicitement la région dans votre code plutôt que de vous fier aux variables d'environnement ou au`config`. Cela rend votre code plus prévisible et moins dépendant de la configuration externe.

Utiliser des variables d'environnement pour le développement et les tests

Pour les environnements de développement et de test, l'utilisation de variables d'environnement permet une plus grande flexibilité sans modifier le code.

Utiliser des profils pour plusieurs environnements

Si votre application doit fonctionner avec plusieurs AWS environnements, pensez à utiliser différents profils dans votre AWS `config` fichier et à passer de l'un à l'autre selon vos besoins.

Utilisation des fournisseurs d'informations d'identification de la AWS SDK pour PHP version 3

Pour des informations de référence sur les mécanismes d'identification disponibles pour le AWS SDKs, voir Identifiants et accès dans le guide de référence des outils AWS SDKs et.

> **Important**
>
> Pour des raisons de sécurité, nous vous recommandons vivement de ne pas utiliser le compte root pour AWS y accéder. Reportez-vous toujours aux meilleures pratiques de sécurité d'IAM dans le guide de l'utilisateur IAM pour connaître les dernières recommandations en matière de sécurité.

Le rôle d'un fournisseur d'informations d'identification dans la AWS SDK pour PHP version 3 est de rechercher et de fournir des informations d'identification aux Service AWS clients du SDK. Le SDK utilise les informations d'identification qu'il obtient pour s'authentifier auprès du service en signant cryptographiquement chaque demande. Les informations d'identification se composent généralement de clés d'accès : un identifiant de clé d'accès et une clé d'accès secrète combinés.

Lorsque vous utilisez des informations d'identification temporaires, par exemple lorsque vous configurez l'authentification IAM Identity Center ou que vous configurez votre environnement d'exécution pour qu'il assume un rôle IAM, un jeton de session est ajouté aux clés d'accès, fournissant un accès limité dans le temps aux ressources. AWS

Qu'est-ce qu'un fournisseur d'informations d'identification dans la AWS SDK pour PHP version 3 ?

Un fournisseur d'informations d'identification est une fonction qui renvoie un `GuzzleHttp\Promise\PromiseInterface` qui est exécutée avec une instance `Aws\Credentials\CredentialsInterface` ou rejetée avec une `Aws\Exception\CredentialsException`. Le SDK fournit plusieurs implémentations de fonctions de fournisseur d'informations d'identification. Vous pouvez également implémenter votre propre logique personnalisée pour créer des informations d'identification ou pour optimiser le chargement des informations d'identification.

Les fournisseurs d'informations d'identification sont transmis à l'option constructeur client `credentials`. Les fournisseurs d'informations d'identification sont asynchrones, ce qui les oblige à être lentement évalués chaque fois qu'une opération d'API est appelée. À ce titre, la transmission

d'une fonction de fournisseur d'informations d'identification à un constructeur client SDK ne valide pas immédiatement les informations d'identification. Si le fournisseur d'informations d'identification ne renvoie pas l'objet des informations d'identification, une opération d'API sera rejetée avec une `Aws\Exception\CredentialsException`.

```
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
]);
```

Comprendre la chaîne de fournisseurs d'informations d'identification par défaut dans la AWS SDK pour PHP version 3

La chaîne de fournisseurs d'informations d'identification par défaut est composée d'une série de fournisseurs d'informations d'identification intégrés invoqués par le SDK. Il est implémenté par la fonction de fournisseur d'informations d'identification DefaultProvider sans paramètres. Une fois les informations d'identification valides trouvées, la recherche s'arrête.

AWS SDK pour PHP Exécute les fournisseurs d'informations d'identification dans l'ordre suivant :

- **`env`**fournisseur : le SDK recherche les clés d'AWS accès définies en tant que variables d'environnement.
- **`assumeRoleWithWebIdentityCredentialProvider`**fournisseur : le SDK recherche les paramètres des fichiers de rôle IAM et de jeton d'identité Web.
- À ce stade de la chaîne, le SDK recherche la configuration dans le partage AWS `config` et `credentials` les fichiers. Le SDK recherche la configuration sous le profil « par défaut », mais si la variable d'`AWS_PROFILE`environnement est définie, le SDK utilise sa valeur de profil nommée.
  - **`sso`**fournisseur : le SDK recherche les paramètres de configuration d'IAM Identity Center dans le fichier partagé`config`.

- **`login_provider`**- Le SDK recherche les paramètres de configuration de la session de connexion à la AWS console dans le `config` fichier partagé.
- **`process`**fournisseur : le SDK recherche le `credential_process` paramètre dans le `credentials` fichier partagé.
- **`ini`**fournisseur : le SDK recherche les AWS informations d'identification ou les informations de rôle IAM dans le fichier partagé`credentials`.
- **`process`**fournisseur : le SDK recherche le `credential_process` paramètre dans le `config` fichier partagé.
- **`ini`**fournisseur : le SDK recherche les AWS informations d'identification ou les informations de rôle IAM dans le fichier partagé`config`.

- **`ecsCredentials`**fournisseur : le SDK recherche les variables d'environnement `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ou les variables fournissant des informations `AWS_CONTAINER_CREDENTIALS_FULL_URI` permettant d'obtenir des informations d'identification temporaires.
- **`instanceProfile`**fournisseur - Le SDK utilise le service de métadonnées d'instance EC2 pour obtenir le rôle IAM spécifié dans le profil d'instance. À l'aide des informations de rôle, le SDK obtient des informations d'identification temporaires.

> **Note**
>
> Le résultat du fournisseur par défaut est automatiquement mémoïsé.

Vous pouvez consulter le code de la chaîne dans le code GitHub source.

Fournisseurs d'informations d'identification intégrés dans la AWS SDK pour PHP version 3

Le SDK fournit plusieurs fournisseurs d'informations d'identification intégrés que vous pouvez utiliser individuellement ou combiner dans une chaîne de fournisseurs d'informations d'identification personnalisée.

Lorsque vous spécifiez un fournisseur d'informations d'identification lors de la création du client de service, le SDK tente de charger les informations d'identification en utilisant uniquement

le fournisseur d'informations d'identification spécifié. Il n'utilise pas la chaîne de fournisseurs d'informations d'identification par défaut. Si vous savez que vous souhaitez qu'un client de service utilise le `instanceProfile` fournisseur, vous pouvez court-circuiter la chaîne par défaut en spécifiant le `instanceProfile` fournisseur dans le constructeur du client de service :

```
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

Les fournisseurs d'informations d'identification sont appelés chaque fois qu'une opération d'API est effectuée. Si le chargement d'informations d'identification est une tâche coûteuse (par exemple, le chargement à partir d'un disque ou d'une ressource réseau) ou si les informations d'identification ne sont pas mises en cache par le fournisseur, envisagez d'encapsuler votre fournisseur d'informations d'identification dans une fonction `Aws\Credentials\CredentialProvider::memoize`. Le fournisseur d'informations d'identification par défaut utilisé par le kit SDK est automatiquement mémoïsé.

Rubriques

- loginfournisseur dans le SDK pour PHP
- assumeRolefournisseur dans le SDK pour PHP
- ssofournisseur dans le SDK pour PHP
- defaultProviderfournisseur dans le SDK pour PHP
- ecsCredentialsfournisseur dans le SDK pour PHP
- envfournisseur dans le SDK pour PHP
- assumeRoleWithWebIdentityCredentialProviderfournisseur dans le SDK pour PHP

- inifournisseur dans le SDK pour PHP
- processfournisseur dans le SDK pour PHP
- instanceProfilefournisseur dans le SDK pour PHP

loginfournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::login`tente de charger les informations d'identification configurées par une session de connexion basée sur un navigateur facilitée par des outils tels que la CLI AWS . Après l'authentification, AWS génère des informations d'identification temporaires qui fonctionnent au niveau local AWS SDKs et dans les outils.

Ce processus vous permet de vous authentifier à l'aide des informations d'identification root créées lors de la configuration initiale du compte, d'un utilisateur IAM ou d'une identité fédérée auprès de votre fournisseur d'identité. Le SDK for AWS PHP gère automatiquement les informations d'identification temporaires pour vous. Cette approche améliore la sécurité en éliminant le besoin de stocker des informations d'identification à long terme localement.

Lorsque vous exécutez la `aws login` commande, vous pouvez effectuer une sélection parmi vos sessions de console actives ou vous connecter via le flux d'authentification basé sur le navigateur, ce qui générera automatiquement des informations d'identification temporaires. Le AWS SDK pour PHP actualisera automatiquement ces informations d'identification, à l'aide du service de connexion, pendant 12 heures maximum.

Le fournisseur de connexion tente de charger le jeton d'accès généré par le flux de travail de session de connexion mentionné précédemment, en fonction du profil fourni. Si aucun profil n'est fourni lors de l'appel du fournisseur, celui-ci tentera de résoudre un profil en vérifiant d'abord la variable d'`AWS_PROFILE`environnement, avant de revenir au profil`default`. La configuration en code peut être transmise au fournisseur, qui recherchera une `region` valeur pour le client du service de connexion utilisé pour actualiser les informations d'identification. Si aucune région n'est fournie dans le tableau de configuration, le fournisseur tentera de résoudre une région en vérifiant la variable d'`AWS_REGION`environnement, puis une valeur de région définie dans le profil résolu. Si aucune région n'est trouvée, le fournisseur renvoie une promesse rejetée avec des instructions sur la façon de configurer une région.

Le fournisseur est appelé dans le cadre de la chaîne par défaut et peut être appelé directement.

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

```
$provider = CredentialProvider::login(<profile_name>, ['region' => <region>]);
// 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',
    'credentials' => $provider
]);
```

Par défaut, si aucune configuration d'informations d'identification n'est fournie sur le client de service que vous souhaitez utiliser, ce fournisseur sera appelé dans le cadre de la chaîne `defaultProvider()` d'informations d'identification. Dans ce scénario, la région du client de service est automatiquement transmise au `login()` fournisseur. Dans ce scénario également, la valeur de profil transmise au fournisseur de connexion sera résolue en vérifiant la variable d'`AWS_PROFILE`environnement, avant de revenir au profil`default`.

`assumeRole`fournisseur dans le SDK pour PHP

Si vous utilisez `Aws\Credentials\AssumeRoleCredentialProvider` pour créer des informations d'identification en assumant un rôle, vous devez fournir les informations `'client'` avec un objet `StsClient` et des détails `'assume_role_params'` comme illustré.

> **Note**
>
> Pour éviter de récupérer inutilement les informations AWS STS d'identification à chaque opération d'API, vous pouvez utiliser cette `memoize` fonction pour gérer l'actualisation automatique des informations d'identification lorsqu'elles expirent. Consultez le code suivant pour obtenir un exemple.

```
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
]);
```

Pour plus d'informations concernant `'assume_role_params'`, voir AssumeRole.

ssofournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::sso`est le fournisseur d'identifiants de connexion unique. Ce fournisseur est également connu sous le nom de fournisseur d' AWS IAM Identity Center informations d'identification.

```
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
```

```
]);
```

Si vous utilisez un profil nommé, remplacez le nom de votre profil par « `default` » dans l'exemple précédent. Pour en savoir plus sur la configuration de profils nommés, consultez la section Shared `config` and `credentials` files dans le guide de référence AWS SDKs and Tools. Vous pouvez également utiliser la variable d'`AWS_PROFILE`environnement pour spécifier les paramètres du profil à utiliser.

Pour en savoir plus sur le fonctionnement du fournisseur IAM Identity Center, consultez la section Comprendre l'authentification IAM Identity Center dans le guide de référence des outils AWS SDKs et.

`defaultProvider`fournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::defaultProvider`est le fournisseur d'informations d'identification par défaut et est également appelé chaîne de fournisseurs d'informations d'identification par défaut. Ce fournisseur est utilisé si vous omettez une option `credentials` lors de la création d'un client. Par exemple, si vous créez un S3Client comme indiqué dans l'extrait suivant, le SDK utilise le fournisseur par défaut :

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

Vous pouvez également utiliser le DefaultProvider dans le code si vous souhaitez fournir des paramètres à des fournisseurs d'informations d'identification spécifiques de la chaîne. Par exemple, l'exemple suivant fournit des paramètres personnalisés de délai d'expiration de connexion et de nouvelle tentative si la fonction `ecsCredentials` fournisseur est utilisée.

```
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
```

```
]);
```

ecsCredentialsfournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::ecsCredentials` tente de charger les informations d'identification via une demande `GET`, dont l'URI est spécifié par la variable d'environnement `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` dans le conteneur.

```
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
]);
```

envfournisseur dans le SDK pour PHP

L'utilisation de variables d'environnement pour contenir vos informations d'identification vous empêche de partager accidentellement votre clé d'accès AWS secrète. Nous vous recommandons de ne jamais ajouter vos clés AWS d'accès directement au client dans les fichiers de production.

Pour vous authentifier auprès d'Amazon Web Services, le SDK vérifie d'abord les informations d'identification dans vos variables d'environnement. Le kit SDK utilise la fonction `getenv()` pour rechercher les variables d'environnement `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` et `AWS_SESSION_TOKEN`. Ces informations d'identification sont appelées informations d'identification d'environnement. Pour savoir comment obtenir ces valeurs, voir Authentifier à l'aide d'informations d'identification à court terme dans le guide de référence AWS SDKs and Tools.

Si vous hébergez votre application sur AWS Elastic Beanstalk, vous pouvez définir les variables d'`AWS_SESSION_TOKEN`environnement `AWS_ACCESS_KEY_ID``AWS_SECRET_KEY`, et via la AWS Elastic Beanstalk console afin que le SDK puisse utiliser ces informations d'identification automatiquement.

Pour plus d'informations sur la définition des variables d'environnement, consultez la section Prise en charge des variables d'environnement dans le Guide de référence des outils AWS SDKs et. De plus,

pour une liste de toutes les variables d'environnement prises en charge par la plupart AWS SDKs, voir Liste des variables d'environnement.

Vous pouvez également définir les variables d'environnement dans la ligne de commande, comme indiqué ici.

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your Compte AWS.
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your Compte AWS.
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your Compte 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 Compte AWS.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your Compte AWS.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your Compte 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` tente de charger les informations d'identification à partir des variables d'environnement.

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

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

`assumeRoleWithWebIdentityCredentialProvider` fournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider` tente de charger les informations d'identification en endossant un rôle. Si les variables d'environnement `AWS_ROLE_ARN` et `AWS_WEB_IDENTITY_TOKEN_FILE` sont présentes, le fournisseur tentera d'endosser le rôle spécifié à `AWS_ROLE_ARN` à l'aide du jeton sur le disque au chemin d'accès complet spécifié dans `AWS_WEB_IDENTITY_TOKEN_FILE`. Si les variables d'environnement sont utilisées, le fournisseur tentera de définir la session à partir de la variable d'environnement `AWS_ROLE_SESSION_NAME`.

Si les variables d'environnement ne sont pas définies, le fournisseur utilise le profil par défaut, ou celui défini en tant que `AWS_PROFILE`. Le fournisseur lit les profils à partir de `~/.aws/credentials` et `~/.aws/config` par défaut, et peut lire depuis des profils spécifiés dans l'option de configuration `filename`. Le fournisseur va endosser le rôle dans `role_arn` du profil, la lecture d'un jeton envoyé à partir du chemin d'accès complet défini dans `web_identity_token_file`. `role_session_name` sera utilisée si elle est définie sur le profil.

Le fournisseur est appelé dans le cadre de la chaîne par défaut et peut être appelé directement.

```
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
]);
```

Par défaut, ce fournisseur d'informations d'identification héritera de la région configurée qui sera utilisée par le StsClient pour assumer le rôle. En option, une version complète StsClient peut être fournie. Les informations d'identification doivent être définies comme `false` sur toutes celles fournies StsClient.

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

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

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

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

inifournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::ini`tente de charger les informations d'identification depuis le partage `config` et `credentials` les fichiers. Par défaut, le SDK tente de charger le profil « par défaut » à partir du AWS `credentials` fichier partagé situé `~/.aws/credentials` dans. Si le SDK trouve la variable d'`AWS_SDK_LOAD_NONDEFAULT_CONFIG`environnement, il recherche également un profil « par défaut » dans le AWS `config` fichier partagé situé à `~/.aws/config` l'adresse.

```
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
]);
```

Vous pouvez utiliser un profil personnalisé ou un emplacement de fichier .ini en fournissant des arguments à la fonction qui crée le fournisseur.

```
$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`fournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::process`tente de charger les informations d'identification en exécutant `credential_process` la valeur spécifiée dans un profil dans un fichier AWS de configuration partagé.

Par défaut, le SDK tente de charger le profil « par défaut » d'abord à partir du AWS `credentials` fichier partagé situé `~/.aws/credentials` dans. Si le profil « par défaut » n'est pas trouvé dans le `credentials` fichier partagé, le SDK recherche le profil par défaut dans le `config` fichier partagé. Voici un exemple de configuration pour le `credentials` fichier partagé.

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

Le SDK appellera la `credential_process` commande exactement comme indiqué en utilisant la `shell_exec` fonction de PHP, puis lira les données JSON depuis stdout. Les informations d'identification `credential_process` doivent être écrites sur stdout au format suivant :

```
{
    "Version": 1,
```

```
    "AccessKeyId": "",
    "SecretAccessKey": "",
    "SessionToken": "",
    "Expiration": ""
}
```

`SessionToken` et `Expiration` sont facultatifs. Le cas échéant, les informations d'identification seront traitées comme temporaires.

```
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
]);
```

Vous pouvez utiliser un profil personnalisé ou un emplacement de fichier .ini en fournissant des arguments à la fonction qui crée le fournisseur.

```
$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`fournisseur dans le SDK pour PHP

`Aws\Credentials\CredentialProvider::instanceProfile`etente de charger les informations d'identification pour un rôle IAM spécifié dans un profil d'instance Amazon EC2.

```
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
]);
```

Par défaut, le fournisseur réessaie de récupérer les informations d'identification jusqu'à trois fois. Le nombre de tentatives peut être défini avec l'`retries`option, et complètement désactivé en définissant l'option sur `0` comme indiqué dans le code suivant.

```
use Aws\Credentials\CredentialProvider;

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

Si la variable d'environnement `AWS_METADATA_SERVICE_NUM_ATTEMPTS` est disponible, sa valeur a priorité sur l'option « réessayer » présentée précédemment.

> **Note**
>
> Vous pouvez désactiver cette tentative de chargement à partir des profils d'instance Amazon EC2 en définissant la variable d'`AWS_EC2_METADATA_DISABLED`Environnement sur. `true`

Enchaînement des fournisseurs d'identifiants dans le SDK for PHP

Vous pouvez chaîner des fournisseurs d'informations d'identification à l'aide de la fonction `Aws\Credentials\CredentialProvider::chain()`. Cette fonction accepte un nombre d'arguments variadique, chacun d'entre eux étant une fonction de fournisseur d'informations d'identification. Cette fonction renvoie ensuite une nouvelle fonction qui est la composition des

fonctions fournies, de façon à ce qu'elles soient appelées les unes après les autres jusqu'à ce que l'un des fournisseurs renvoie une promesse tenue.

Le `defaultProvider` utilise cette composition pour vérifier plusieurs fournisseurs avant un échec. La source du `defaultProvider` illustre l'utilisation de la fonction `chain`.

```
// 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)
        )
    );
}
```

Création d'un fournisseur d'informations d'identification personnalisé à utiliser avec le SDK for PHP

Les fournisseurs d'informations d'identification sont simplement des fonctions qui, lorsqu'elles sont appelées, renvoient une promesse (`GuzzleHttp\Promise\PromiseInterface`) tenue grâce à un objet `Aws\Credentials\CredentialsInterface` ou rejetée via une `Aws\Exception\CredentialsException`.

Une bonne pratique pour créer des fournisseurs consiste à créer une fonction qui est appelée pour créer le véritable fournisseur d'informations d'identification. Par exemple, voici la source du fournisseur `env` (légèrement modifiée pour cet exemple). Notez qu'il s'agit d'une fonction qui renvoie la fonction du véritable fournisseur. Ceci vous permet de composer facilement des fournisseurs d'informations d'identification et de les transmettre comme valeurs.

```
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));
    };
}
```

Mémorisation des informations d'identification dans le SDK for PHP

Il est parfois nécessaire de créer un fournisseur d'informations d'identification qui se souvient de la dernière valeur renvoyée. Ceci peut être utile aux performances lorsque le chargement des informations d'identification est une opération coûteuse ou lors de l'usage de la classe `Aws\Sdk` pour partager un fournisseur d'informations d'identification entre plusieurs clients. Vous pouvez ajouter la mémoïsation à un fournisseur d'informations d'identification en encapsulant la fonction du fournisseur d'informations d'identification dans une fonction `memoize`.

```
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());
```

Lorsque les informations d'identification mémoïsées expirent, l'habillage mémoïsé appelle le fournisseur habillé dans une tentative de rafraîchissement des informations d'identification.

Assumez un rôle IAM à l'aide de la AWS SDK pour PHP version 3

Utilisation de rôles IAM pour les informations d'identification variables d'instance Amazon EC2

Si vous exécutez votre application sur une instance Amazon EC2, la méthode préférée pour fournir des informations d'identification pour les appels AWS consiste à utiliser un rôle IAM pour obtenir des informations de sécurité temporaires.

Lorsque vous utilisez des rôles IAM, vous n'avez pas à vous soucier de la gestion des informations d'identification depuis votre application. Ils permettent à une instance d' « assumer » un rôle en récupérant des informations d'identification temporaires depuis le serveur de métadonnées de l'instance Amazon EC2.

Les informations d'identification temporaires, souvent appelées informations d'identification du profil d'instance, permettent d'accéder aux actions et aux ressources autorisées par la politique du rôle. Amazon EC2 se charge de l'authentification sécurisée des instances auprès du service IAM pour qu'elles assument le rôle et de l'actualisation périodique des informations d'identification de rôle récupérées. Ceci permet de sécuriser l'application presque sans effort de votre part. Pour une liste des services prenant en charge les informations d’identification de sécurité temporaires, consultez Services AWS qui fonctionnent avec IAM dans le Guide de l’utilisateur IAM.

> **Note**
>
> Pour éviter d'accéder au service de métadonnées chaque fois, vous pouvez transmettre une instance `Aws\CacheInterface` en tant qu'option `'credentials'` à un constructeur client. Ceci permet au kit SDK d'utiliser des informations d'identification du profil d'instance mises en cache à la place. Pour plus de détails, voir Configuration pour la AWS SDK pour PHP version 3.

Pour plus d'informations sur le développement d'applications Amazon EC2 à l'aide de SDKs, consultez la section Utilisation des rôles IAM pour les instances Amazon EC2 dans le AWS SDKs guide de référence and Tools.

Création et attribution d'un rôle IAM à une instance Amazon EC2

1. Créez un client IAM.

   Importations

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

   use Aws\Iam\IamClient;
   ```

   Exemple de code

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

2. Créez un rôle IAM avec les autorisations nécessaires pour les actions et les ressources que vous utiliserez.

   Exemple de code

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

3. Créez un profil d'instance IAM et stockez le nom de ressource Amazon (ARN) à partir du résultat.

   > **Note**
   >
   > Si vous utilisez la console IAM au lieu de AWS SDK pour PHP, la console crée automatiquement un profil d'instance et lui donne le même nom que le rôle auquel il correspond.
   >
   > Exemple de code
   >
   > ```
   > $IPN = 'InstanceProfileName';
   >
   > $result = $client->createInstanceProfile([
   >     'InstanceProfileName' => $IPN ,
   > ```

```
]);

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

4. Créez un client Amazon EC2.

   Importations

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

   use Aws\Ec2\Ec2Client;
   ```

   Exemple de code

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

5. Ajoutez le profil d'instance à une instance Amazon EC2 en cours d'exécution ou arrêtée. Utilisez le nom du profil d'instance de votre rôle IAM.

   Exemple de code

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

Pour de plus amples informations sur les rôles IAM, veuillez consulter Rôles IAM pour Amazon EC2 dans le Guide de l'utilisateur Amazon EC2.

Utilisation de rôles IAM pour les tâches Amazon ECS

Une tâche dans Amazon Elastic Container Service (Amazon ECS) peut assumer un rôle IAM pour AWS effectuer des appels d'API. Il s'agit d'une stratégie de gestion des informations d'identification à

utiliser par vos applications, similaire à la manière dont les profils d'instance Amazon EC2 fournissent des informations d'identification aux instances Amazon EC2.

Au lieu de créer et de distribuer des AWS informations d'identification à long terme aux conteneurs ou d'utiliser le rôle de l'instance Amazon EC2, vous pouvez associer un rôle IAM qui utilise des informations d'identification temporaires à une définition de tâche ECS ou `RunTask` à une opération d'API.

Pour plus d'informations sur l'utilisation des rôles IAM que les tâches de conteneur peuvent assumer, consultez la rubrique relative au rôle IAM des tâches dans le manuel Amazon ECS Developer Guide. Pour des exemples d'utilisation du rôle IAM de tâche sous la forme d'une `taskRoleArn` définition de tâche, consultez la section Exemples de définitions de tâches également dans le manuel Amazon ECS Developer Guide.

Assumer un rôle IAM dans un autre Compte AWS

Lorsque vous travaillez dans un Compte AWS (compte A) et que vous souhaitez assumer un rôle dans un autre compte (compte B), vous devez d'abord créer un rôle IAM dans le compte B. Ce rôle permet aux entités de votre compte (compte A) d'effectuer des actions spécifiques dans le compte B. Pour plus d'informations sur l'accès entre comptes, voir Tutoriel : déléguer l'accès entre AWS comptes à l'aide de rôles IAM.

Une fois que vous avez créé un rôle dans le compte B, enregistrez l'ARN de rôle. Vous utiliserez cet ARN lorsque vous assumerez le rôle depuis le compte A. Vous assumez le rôle en utilisant les AWS informations d'identification associées à votre entité dans le compte A.

Créez un AWS STS client avec des informations d'identification pour votre Compte AWS. Dans ce qui suit, nous avons utilisé un profil d'informations d'identification, mais vous pouvez utiliser n'importe quelle méthode. Avec le client AWS STS que vous venez de créer, appelez la commande assume-role et fournissez un élément sessionName personnalisé. Récupérez les nouvelles informations d'identification temporaires à partir du résultat. Les informations d'identification durent une heure par défaut.

Exemple de code

```
$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']
    ]
]);
```

Pour plus d'informations, consultez la section Utilisation des rôles IAM ou consultez le Guide AssumeRolede référence de l' AWS SDK pour PHP API.

Utilisation d'un rôle IAM avec une identité Web

La Fédération des identités Web permet aux clients d'utiliser des fournisseurs d'identité tiers pour s'authentifier lorsqu'ils accèdent aux AWS ressources. Pour pouvoir assumer un rôle avec la fédération d'identité web, vous devez créer un rôle IAM et configurer un fournisseur d'identité web (IdP). Pour de plus amples informations, veuillez consulter Création d'un rôle pour la fédération d'identité web ou OpenID Connect Federation (Console).

Après avoir créé un fournisseur d'identité et créé un rôle pour votre identité Web, utilisez un AWS STS client pour authentifier un utilisateur. Indiquez le webIdentityToken et ProviderId pour votre identité, et l'ARN du rôle IAM avec des autorisations pour l'utilisateur.

Exemple de code

```
$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']
    ]
]);
```

Pour plus d'informations, consultez AssumeRoleWithWebIdentity—Federation via un fournisseur d'identité basé sur le Web ou AssumeRoleWithWebIdentitydans le manuel de référence des AWS SDK pour PHP API.

Assumer un rôle avec un profil

Définissez les profils dans **`~/.aws/credentials`**

Vous pouvez configurer le AWS SDK pour PHP pour utiliser un rôle IAM en définissant un profil dans`~/.aws/credentials`.

Créez un nouveau profil avec le `role_arn` paramètre correspondant au rôle que vous souhaitez assumer. Incluez également le `source_profile` paramètre d'un autre profil avec des informations d'identification autorisées à assumer le rôle IAM. Pour plus de détails sur ces paramètres de configuration, consultez la section Assumer les informations d'identification du rôle dans le guide de référence AWS SDKs et Tools.

Par exemple, dans ce qui suit`~/.aws/credentials`, le `project1` profil définit `role_arn` et spécifie le `default` profil comme source des informations d'identification afin de vérifier que l'entité qui leur est associée peut assumer le rôle.

```
[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
```

Si vous définissez la variable d'`AWS_PROFILE`environnement ou si vous utilisez un `profile` paramètre lorsque vous instanciez un client de service, le rôle spécifié dans `project1` est assumé, en utilisant le `default` profil comme informations d'identification source.

L'extrait suivant montre l'utilisation du `profile` paramètre dans un `S3Client` constructeur. Ils `S3Client` auront les autorisations associées au rôle associé au `project1` profil.

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

Définissez les profils dans **`~/.aws/config`**

Le `~/.aws/config` fichier peut également contenir les profils que vous souhaitez faire passer pour acquis. Si vous définissez la variable d'environnement`AWS_SDK_LOAD_NONDEFAULT_CONFIG`, le SDK for PHP charge les profils depuis `config` le fichier. Lorsque cette `AWS_SDK_LOAD_NONDEFAULT_CONFIG` option est définie, le SDK charge les profils à la fois depuis `~/.aws/config` et`~/.aws/credentials`. Les profils de `~/.aws/credentials` sont chargés en dernier et ont priorité sur un profil du `~/.aws/config` même nom. Les profils définis dans ces emplacements peuvent être utilisés en tant que `source_profile` ou que profil à assumer.

L'exemple suivant utilise le `project1` profil défini dans le `config` fichier et le `default` profil dans le `credentials` fichier. Le `AWS_SDK_LOAD_NONDEFAULT_CONFIG` est également défini.

```
# 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
```

Lorsque le `S3Client` constructeur exécute l'extrait de code suivant, le rôle défini dans le `project1` profil sera assumé à l'aide des informations d'identification associées au profil. `default`

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

Utiliser des informations d'identification temporaires depuis AWS STS le SDK for PHP

AWS Security Token Service (AWS STS) vous permet de demander des privilèges limités, des informations d'identification temporaires pour les utilisateurs IAM ou pour les utilisateurs que vous authentifiez via la fédération d'identité. Pour une meilleure compréhension, consultez la section Informations d'identification de sécurité temporaires dans le guide de l'utilisateur IAM. Vous pouvez utiliser des informations d'identification de sécurité temporaires pour accéder à la plupart AWS des services. Pour une liste des services prenant en charge les informations d'identification de sécurité temporaires, consultez Services AWS qui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

L'un des cas d'utilisation courants des informations d'identification temporaires consiste à accorder aux applications mobiles ou côté client l'accès aux AWS ressources en authentifiant les utilisateurs par le biais de fournisseurs d'identité tiers (voir Fédération des identités Web).

Obtenir des informations d'identification temporaires

AWS STS comporte plusieurs opérations qui renvoient des informations d'identification temporaires, mais l'`GetSessionToken`opération est la plus simple à démontrer. L'extrait suivant récupère les

informations d'identification temporaires en appelant la `getSessionToken` méthode du client STS du SDK PHP.

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

$stsClient = $sdk->createSts();

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

Le résultat pour `GetSessionToken` et les autres AWS STS opérations contient toujours une `'Credentials'` valeur. Si vous imprimez le `$result` (par exemple en utilisant`print_r($result)`), cela ressemble à ce qui suit.

```
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>'
    )
    ...
)
```

Fournir des informations d'identification temporaires au AWS SDK pour PHP

Vous pouvez utiliser des informations d'identification temporaires avec un autre AWS client en instanciant le client et en lui transmettant les valeurs reçues AWS STS directement.

```
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']
    ]
]);
```

Vous pouvez également construire un objet `Aws\Credentials\Credentials` et l'utiliser lorsque vous instanciez le client.

```
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
]);
```

Toutefois, le meilleur moyen de fournir des informations d'identification temporaires consiste à utiliser la méthode d'assistance `createCredentials()` incluse avec le `StsClient`. Cette méthode extrait les données d'un AWS STS résultat et crée l'`Credentials`objet pour vous.

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

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

Pour plus d'informations sur les raisons pour lesquelles vous pourriez avoir besoin d'utiliser des informations d'identification temporaires dans votre application ou votre projet, consultez la section Scénarios d'octroi d'un accès temporaire dans la AWS STS documentation.

Créez des clients anonymes dans le SDK for PHP

Dans certains cas, il se peut que vous souhaitiez créer un client qui n'est associé à aucune information d'identification. Ceci vous permet d'effectuer des demandes anonymes à un service.

Par exemple, vous pouvez configurer à la fois des objets Amazon S3 et des CloudSearch domaines Amazon pour autoriser un accès anonyme.

Pour créer un client anonyme, vous pouvez définir l'option `'credentials'` sur `false`.

```
$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',
]);
```

Utilisez la commande AWS Temps d'exécution commun (AWS extension CRT) dans le SDK pour PHP

Les bibliothèques AWS CRT fournissent des fonctionnalités de base avec de bonnes performances et un encombrement minimal pour plusieurs AWS SDK. Cette rubrique explique quand le AWS CRT est utilisé par le SDK pour PHP et comment installer l'extension AWS CRT.

Lorsque vous avez besoin de AWS Extension CRT installée

Le SDK pour PHP utilise les fonctionnalités d'autorisation et de somme de contrôle des bibliothèques AWS CRT. L'extension AWS CRT est requise lorsque vous travaillez avec :

- Points d' Multi-Region accès Amazon S3
- Points de terminaison EventBridge mondiaux Amazon
- Algorithme de CRC-32C somme de contrôle dans Amazon Simple Storage Service (Amazon S3)

Si vous utilisez l'une des fonctionnalités répertoriées ci-dessus et que l'extension AWS CRT n'est pas installée dans votre environnement PHP, le SDK pour PHP signalera un message d'erreur et vous rappellera d'installer l'extension.

Installer la AWS Temps d'exécution commun (AWS Rallonge (CRT)

Les instructions relatives à l'installation de l'extension AWS CRT sont disponibles sur la page principale du GitHub référentiel aws-crt-php.

Utilisation de la AWS SDK pour PHP version 3

Ce chapitre fournit des conseils complets sur l'utilisation de la AWS SDK pour PHP version 3 pour interagir Services AWS efficacement avec. Vous apprendrez les techniques essentielles pour effectuer des demandes de service, gérer les erreurs et tirer parti des fonctionnalités avancées du SDK pour créer des applications PHP robustes. AWS

Que vous développiez une application simple ou un système complexe, les techniques décrites dans ce chapitre vous aideront à optimiser votre code, à améliorer la gestion des erreurs et à implémenter des modèles efficaces à Services AWS utiliser via le SDK PHP.

Rubriques

- Faire des Service AWS demandes à l'aide de la AWS SDK pour PHP version 3
- Programmation asynchrone à l'aide de la version 3 AWS SDK pour PHP
- Gestion des erreurs dans la AWS SDK pour PHP version 3
- Gestionnaires et intergiciels dans la version 3 AWS SDK pour PHP
- Streams dans la AWS SDK pour PHP version 3
- Utilisation de résultats paginés dans la AWS SDK pour PHP version 3
- Serveurs dans la AWS SDK pour PHP version 3
- JMESPath expressions dans la AWS SDK pour PHP version 3

Faire des Service AWS demandes à l'aide de la AWS SDK pour PHP version 3

Vue d'ensemble du flux de travail des demandes de SDK

L'utilisation de la AWS SDK pour PHP version 3 suit un schéma cohérent dans l'ensemble Services AWS. Le flux de travail de base comprend trois étapes principales :

1. Créez un client de service : instanciez un objet client pour celui que Service AWS vous souhaitez utiliser.
2. Exécuter des opérations : appelez des méthodes sur le client qui correspondent aux opérations de l'API du service.

3. Résultats du traitement : utilisez l'objet Result de type tableau renvoyé en cas de succès, ou gérez toute exception déclenchée en cas d'échec.

Les sections suivantes expliquent chacune de ces étapes en détail, en commençant par la procédure de création et de configuration des clients de service.

Création d'un client de service de base

Vous pouvez créer un client en transmettant un tableau associatif d'options au constructeur d'un client.

Importations

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

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

Exemple de code

```
//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'.
```

Les informations relatives au paramètre optionnel « version » sont disponibles dans la rubrique des options de configuration.

Notez que nous n'avons pas explicitement fourni d'informations d'identification au client. En effet, le SDK utilise la chaîne de fournisseurs d'informations d'identification par défaut pour rechercher les informations d'identification.

Toutes les options générales de configuration du client sont décrites en détail dansOptions du constructeur client pour la AWS SDK pour PHP version 3. Les options fournies peuvent varier selon le type de client que vous créez. Ces options de configuration client personnalisées sont décrites dans la documentation sur l'API de chaque client.

Bien que l'exemple ci-dessus montre la création de clients de base, vous pouvez personnaliser vos clients de service pour répondre à des exigences spécifiques. Pour des informations plus détaillées

sur la configuration des clients de service par le biais du code, consultezConfiguration des clients de service dans le code pour la AWS SDK pour PHP version 3. Si vous devez configurer des clients de service à l'aide de fichiers de configuration externes ou de variables d'environnement, reportez-vous àConfiguration externe des clients de service pour la AWS SDK pour PHP version 3.

Demandes

Vous pouvez effectuer des demandes de service en appelant la méthode du même nom sur un objet client. Par exemple, pour effectuer l'PutObjectopération Amazon S3, vous appelez la `Aws\S3\S3Client::putObject()` méthode.

Importations

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

use Aws\S3\S3Client;
```

Exemple de code

```
// 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' => 'amzn-s3-demo-bucket',
    'Key' => 'my-key',
    'Body' => 'this is the body!'
]);

// Download the contents of the object.
$result = $s3Client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
```

```
    'Key' => 'my-key'
]);

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

Les opérations disponibles pour un client, ainsi que la structure de l'entrée et de la sortie, sont définies au moment de l'exécution en fonction d'un fichier de description du service. Lorsque vous créez un client, si vous ne fournissez aucun `version` paramètre (par exemple, « 01/03/2006 » ou « dernier ») du modèle de service, le client utilise par défaut la dernière version. Le kit SDK détecte le fichier de configuration correspondant en fonction de la version fournie.

Toutes les méthodes d'opération telles que `putObject()` acceptent un seul argument : un tableau associatif qui représente les paramètres de l'opération. La structure de ce tableau (et la structure de l'objet de résultat) est définie pour chaque opération dans la documentation sur l'API du SDK (par exemple, consultez les documents sur l'API pour de plus amples informations sur l'opération putObject).

Options du gestionnaire HTTP

Vous pouvez également ajuster la manière dont le gestionnaire HTTP sous-jacent exécute la requête à l'aide du paramètre spécial `@http`. Vous pouvez inclure dans le paramètre `@http` les mêmes options que celles définies lors de l'instanciation du client à l'aide de l'option client « http ».

```
// 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'
    ]
]);
```

Utilisation des objets de résultat

L'exécution d'une opération réussie renvoie un objet `Aws\Result`. Plutôt que de renvoyer des données brutes XML ou JSON d'un service, le kit SDK renseigne les données de réponse dans une structure de tableau associatif. Cette approche normalise certains aspects des données en fonction des connaissances du service spécifique et de la structure de réponse sous-jacente.

Vous pouvez accéder aux données de l'`AWS\Result`objet sous la forme d'un tableau PHP associatif.

Importations

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

Exemple de code

```
// 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();
```

Le contenu de l'objet de résultat dépend de l'opération qui a été exécutée et de la version du service. La structure du résultat de chaque opération d'API est détaillée dans la documentation sur l'API.

Le SDK est intégré à JMESPathun DSL utilisé pour rechercher et manipuler des données JSON ou, dans notre cas, des tableaux PHP. L'objet de résultat contient une méthode `search()` que vous pouvez utiliser pour extraire des données par déclaration du résultat.

Exemple de code

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

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

Objets de commande dans la AWS SDK pour PHP version 3

AWS SDK pour PHP Utilise le modèle de commande pour encapsuler les paramètres et le gestionnaire qui seront utilisés pour transférer une requête HTTP ultérieurement.

Utilisation implicite de commandes

Si vous examinez une classe client, vous constaterez que les méthodes correspondant aux opérations d'API n'existent pas réellement. Elles sont implémentées à l'aide de la méthode magique `__call()`. Ces pseudo-méthodes sont en réalité des raccourcis qui encapsulent l'utilisation des objets de commande par le kit SDK.

Vous n'avez généralement pas besoin d'interagir directement avec les objets de commande. Lorsque vous appelez des méthodes telles que `Aws\S3\S3Client::putObject()`, le kit SDK crée un objet `Aws\CommandInterface` en fonction des paramètres fournis, exécute la commande et renvoie un objet `Aws\ResultInterface` renseigné (ou lève une exception en cas d'erreur). Un flux similaire se produit lorsqu'une des méthodes `Async` d'un client (par exemple, `Aws\S3\S3Client::putObjectAsync()`) est appelée : le client crée une commande en fonction des paramètres fournis, sérialise une demande HTTP, initie la demande et renvoie une promesse.

Les exemples suivants sont équivalents d'un point de vue fonctionnel.

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

$params = [
    'Bucket' => 'amzn-s3-demo-bucket',
    '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);
```

Paramètres de commande

Toutes les commandes prennent en charge un certain nombre de paramètres spéciaux qui ne font pas partie de l'API d'un service, mais contrôlent le comportement du kit SDK.

`@http`

Vous pouvez, à l'aide de ce paramètre, ajuster la manière dont le gestionnaire HTTP sous-jacent exécute la demande. Vous pouvez inclure dans le paramètre `@http` les mêmes options que celles définies lors de l'instanciation du client à l'aide de l'option client « http ».

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

`@retries`

Comme l'option client « retries », `@retries` (nouvelles tentatives) contrôle le nombre de fois où une commande peut être relancée avant que celle-ci ne soit considérée comme ayant échoué. Définissez-la sur `0` pour désactiver les nouvelles tentatives.

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

> **Note**
>
> Si vous avez désactivé les nouvelles tentatives sur un client, vous ne pouvez pas les activer de manière sélective sur des commandes individuelles transmises à ce client.

Création d'objets de commande

Vous pouvez créer une commande à l'aide de la méthode `getCommand()` d'un client. Une demande HTTP ne sera pas immédiatement exécutée ou transférée. L'exécution de la demande ne sera effectuée qu'une fois la commande transmise à la méthode `execute()` du client. Vous avez ainsi la possibilité de modifier l'objet de commande avant d'exécuter la commande.

```
$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);
```

Commande **`HandlerList`**

Lorsqu'une commande est créée à partir d'un client, elle reçoit un clone de l'objet `Aws\HandlerList` du client. La commande reçoit un clone de la liste des gestionnaires du client pour autoriser une commande à utiliser un intergiciel et des gestionnaires personnalisés qui n'affectent pas les autres commandes exécutées par le client.

Cela signifie que vous pouvez utiliser un client HTTP différent par commande (par exemple, `Aws\MockHandler`) et ajouter un comportement personnalisé par commande via l'intergiciel. L'exemple suivant utilise un `MockHandler` pour créer des résultats fictifs au lieu d'envoyer des demandes HTTP réelles.

```
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'
```

En plus de changer le gestionnaire utilisé par la commande, vous pouvez également injecter un intergiciel personnalisé dans la commande. L'exemple suivant utilise l'intergiciel `tap`, qui fonctionne comme observateur dans la liste des gestionnaires.

```
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`

Le `Aws\CommandPool` vous permet d'exécuter des commandes simultanément à l'aide d'un itérateur générant des objets `Aws\CommandInterface`. Le `CommandPool` veille à ce qu'un nombre constant de commandes soient exécutées simultanément pendant l'itération sur les commandes du groupe (à mesure que les commandes se terminent, d'autres sont exécutées pour garantir une taille de groupe constante).

Voici un exemple très simple illustrant l'envoi de quelques commandes à l'aide d'un `CommandPool`.

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

// Create the client
$client = new S3Client([
```

```
    'region'  => 'us-standard',
    'version' => '2006-03-01'
]);

$bucket = 'amzn-s3-demo-bucket';
$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();
```

Cet exemple est relativement peu puissant pour le `CommandPool`. Essayons un exemple plus complexe. Supposons que vous souhaitiez télécharger des fichiers sur disque dans un compartiment Amazon S3. Pour obtenir une liste des fichiers à partir du disque, nous pouvons utiliser le `DirectoryIterator` de PHP. Cet itérateur génère des objets `SplFileInfo`. Le `CommandPool` accepte un itérateur générant des objets `Aws\CommandInterface`. Nous devons donc mapper les objets `SplFileInfo` pour renvoyer des objets `Aws\CommandInterface`.

```
<?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"; });
```

Configuration de l'**`CommandPool`**

Le constructeur `Aws\CommandPool` accepte différentes options de configuration.

concurrency (callable|int)

> Nombre maximal de commandes pouvant être exécutées simultanément. Fournissez une fonction pour redimensionner le groupe de manière dynamique. La fonction reçoit le nombre actuel de demandes en attente et devrait renvoyer un nombre entier représentant la nouvelle limite de taille du groupe.

before (callable)

> Fonction à appeler avant d'envoyer chaque commande. La fonction `before` accepte la commande et la clé de l'itérateur de la commande. Vous pouvez muter la commande, si nécessaire, dans la fonction `before` avant d'envoyer la commande.

fulfilled (callable)

> Fonction à appeler lorsqu'une promesse est exécutée. La fonction reçoit l'objet de résultat, l'ID de l'itérateur à partir duquel provient le résultat et la promesse agrégée pouvant être résolue ou rejetée si vous devez court-circuiter le groupe.

rejected (callable)

Fonction à appeler lorsqu'une promesse est rejetée. La fonction reçoit un objet `Aws\Exception`, l'ID de l'itérateur à partir duquel provient l'exception et la promesse agrégée pouvant être résolue ou rejetée si vous devez court-circuiter le groupe.

Collecte manuelle des déchets entre les commandes

Si vous atteignez la limite de mémoire lors de groupes de commandes volumineux, c'est peut-être parce que des références cycliques générées par le kit SDK n'ont pas encore été traitées par le nettoyage de mémoire de PHP. L'appel manuel de l'algorithme de nettoyage entre les commandes peut permettre le traitement des cycles avant que cette limite ne soit atteinte. L'exemple suivant crée un `CommandPool` qui appelle l'algorithme de nettoyage en utilisant un rappel avant l'envoi de chaque commande. Notez que l'appel du nettoyage de mémoire représente un coût en termes de performances, et que l'utilisation optimale dépendra de votre cas d'utilisation et de l'environnement.

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

Programmation asynchrone à l'aide de la version 3 AWS SDK pour PHP

Vous pouvez envoyer des commandes simultanément à l'aide des fonctions asynchrones du kit SDK. Vous pouvez envoyer des requêtes de manière asynchrone en ajoutant `Async` à la fin du nom d'une opération. Cette méthode lance la requête et renvoie une promesse.

La promesse est exécutée avec l'objet de résultat en cas de réussite ou rejetée avec une exception en cas d'échec. Cela vous permet de créer plusieurs promesses et de leur faire envoyer des requêtes HTTP simultanément lorsque le gestionnaire HTTP sous-jacent transfère les requêtes.

Importations

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

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

Exemple de code

```
// 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();
```

Vous pouvez forcer une promesse à exécuter les opérations de manière synchrone à l'aide de la méthode `wait` de la promesse. L'exécution forcée de la promesse « débloque » l'état de la promesse par défaut : le résultat de la promesse est renvoyé ou l'exception rencontrée est levée. Lorsque vous appelez `wait()` sur une promesse, le processus se bloque jusqu'à ce que la requête HTTP soit terminée et que le résultat soit renseigné ou qu'une exception soit levée.

Lorsque vous utilisez le kit SDK avec une bibliothèque de boucles d'événements, ne bloquez pas les résultats. Utilisez plutôt la méthode `then()` d'un résultat pour accéder à une promesse résolue ou rejetée une fois l'opération terminée.

Importations

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

Exemple de code

```
// 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();
    });
```

Promesses de la AWS SDK pour PHP version 3

Les AWS SDK pour PHP utilisations promettent de permettre des flux de travail asynchrones, et cette asynchronicité permet d'envoyer des requêtes HTTP simultanément. La spécification de promesse utilisée par le kit SDK est Promesses/A+.

Qu'est-ce qu'une promesse ?

Une promesse représente le résultat final d'une opération asynchrone. Le principal moyen d'interagir avec une promesse est via sa méthode `then`. Cette méthode enregistre les rappels pour recevoir la valeur finale d'une promesse ou la raison pour laquelle la promesse ne peut pas être exécutée.

Il AWS SDK pour PHP s'appuie sur le package guzzlehttp/promises Composer pour son implémentation des promesses. Les promesses Guzzle prennent en charge les flux de travail bloquants et non bloquants, et peuvent être utilisées avec n'importe quelle boucle d'événements non bloquants.

> **Note**
>
> Les requêtes HTTP sont envoyées simultanément en AWS SDK pour PHP utilisant un seul thread, dans lequel des appels non bloquants sont utilisés pour transférer une ou plusieurs requêtes HTTP tout en réagissant aux changements d'état (par exemple, remplir ou rejeter des promesses).

Promesses dans le kit SDK

Les promesses sont utilisées dans l'ensemble du kit SDK. Par exemple, les promesses sont utilisées dans la plupart des abstractions de haut niveau fournies par le SDK : paginateurs, serveurs, pools de commandes, téléchargements partitionnés, transferts S3, etc. directory/bucket

Tous les clients fournis par le kit SDK renvoient des promesses lorsque vous appelez l'une des méthodes suffixées `Async`. Par exemple, le code suivant montre comment créer une promesse pour obtenir les résultats d'une opération Amazon `DescribeTable` DynamoDB.

```
$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']);
```

Notez que vous pouvez appeler `describeTable` ou `describeTableAsync`. Ces méthodes sont des méthodes `__call` magiques utilisées sur un client et alimentées par le modèle d'API et le numéro `version` associé au client. En appelant des méthodes telles que `describeTable` sans le suffixe `Async`, le client se bloquera pendant l'envoi d'une demande HTTP et renverra un objet `Aws\ResultInterface` ou lèvera une `Aws\Exception\AwsException`. En ajoutant le suffixe `Async` au nom de l'opération (c.-à-d. `describeTableAsync`), le client créera une promesse qui sera exécutée avec un objet `Aws\ResultInterface` ou rejetée avec un `Aws\Exception\AwsException`.

> **Important**
>
> Lorsque la promesse est renvoyée, le résultat est peut-être déjà arrivé (par exemple, lors de l'utilisation d'un gestionnaire fictif), ou la demande HTTP n'a peut-être pas été initiée.

Vous pouvez enregistrer un rappel avec la promesse en utilisant la méthode `then`. Cette méthode accepte deux rappels, `$onFulfilled` et `$onRejected`, tous deux facultatifs. Le rappel `$onFulfilled` est appelé si la promesse est exécutée, et le rappel `$onRejected` est appelé si la promesse est rejetée (c.-à-d. en cas d'échec).

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

```
);
```

Exécution simultanée de commandes

Plusieurs promesses peuvent être composées conjointement de manière à être exécutées simultanément. Cette action peut être réalisée en intégrant le kit SDK à une boucle d'événements non bloquants, ou en générant plusieurs promesses et en attendant qu'elles se terminent simultanément.

```
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**
>
> CommandPoolIl fournit un mécanisme plus puissant pour exécuter plusieurs opérations d'API simultanément.

Enchaîner des promesses

L'un des principaux avantages des promesses est qu'elles sont composables, vous permettant ainsi de créer des pipelines de transformation. Les promesses sont composées en créant des chaînes de

rappels `then`, suivis de rappels `then`. La valeur de retour d'une méthode `then` est une promesse qui est exécutée ou rejetée en fonction du résultat des rappels fournis.

```
$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**
>
> La valeur de retour d'un rappel de promesse est l'argument `$value` fourni aux promesses en aval. Si vous souhaitez fournir une valeur aux chaînes de promesses en aval, vous devez renvoyer une valeur dans la fonction de rappel.

Transfert des refus

Vous pouvez enregistrer un rappel à appeler lorsqu'une promesse est rejetée. Si une exception est levée dans un rappel, la promesse est rejetée avec l'exception et les promesses suivantes de la

chaîne sont rejetées avec l'exception. Si une valeur est renvoyée à partir d'un rappel `$onRejected`, les promesses suivantes de la chaîne de promesse sont exécutées avec la valeur de retour du rappel `$onRejected`.

En attente de promesses

Vous pouvez forcer des promesses à s'exécuter de manière synchrone en utilisant la méthode `wait` d'une promesse.

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

Si une exception se produit lors de l'appel de la fonction `wait` d'une promesse, la promesse est rejetée avec l'exception et l'exception est levée.

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

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

L'appel de la méthode `wait` d'une promesse qui a été exécutée ne déclenche pas la fonction wait. Cela renvoie simplement la valeur précédemment fournie.

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

Appeler la méthode `wait` d'une promesse qui a été rejetée lève une exception. Si la raison du rejet est une instance de `\Exception`, la raison est levée. Sinon, une `GuzzleHttp\Promise \RejectionException` est levée et la raison peut être obtenue en appelant la méthode `getReason` de l'exception.

> **Note**
>
> Les appels d'opération d'API dans le AWS SDK pour PHP sont rejetés avec les sous-classes de la `Aws\Exception\AwsException` classe. Toutefois, la raison fournie à une méthode `then` peut être différente en raison de l'ajout d'un intergiciel personnalisé modifiant une raison de rejet.

Annulation de promesses

Les promesses peuvent être annulées à l'aide de la méthode `cancel()` d'une promesse. Si une promesse a déjà été résolue, appeler `cancel()` n'aura aucun effet. L'annulation d'une promesse annule la promesse et toutes les promesses en attente de la livraison de la promesse. Un promesse annulée est rejetée avec une `GuzzleHttp\Promise\RejectionException`.

Combiner les promesses

Vous pouvez combiner des promesses dans des promesses agrégées afin de créer des flux de travail plus sophistiqués. Le package `guzzlehttp/promise` contient différentes fonctions que vous pouvez utiliser pour combiner des promesses.

Vous pouvez trouver la documentation de l'API pour toutes les fonctions de collecte de promesses sur namespace- GuzzleHttp .Promise.

each et each_limit

Utilisez les `Aws\CommandInterface` commandes CommandPoollorsque vous avez une file d'attente de tâches à exécuter simultanément avec une taille de pool fixe (les commandes peuvent être stockées en mémoire ou produites par un itérateur paresseux). Le `CommandPool` veille à ce qu'un nombre fixe de commandes soit envoyées simultanément jusqu'à ce que l'itérateur fourni soit épuisé.

Le `CommandPool` fonctionne uniquement avec les commandes exécutées par le même client. Vous pouvez utiliser la fonction `GuzzleHttp\Promise\each_limit` pour effectuer des commandes d'envoi de différents clients simultanément avec une taille de groupe fixe.

```
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();
```

Coroutines prometteuses

La bibliothèque de promesses Guzzle propose une fonction particulièrement puissante qui autorise l'utilisation des coroutines de promesses rendant l'écriture de flux de travail asynchrones plus semblable à celle de flux de travail synchrones classiques. En effet, le kit AWS SDK pour PHP utilise les promesses de coroutine dans la plupart des abstractions de haut niveau.

Supposons que vous souhaitiez créer plusieurs compartiments, charger un fichier dans le compartiment une fois ce dernier disponible et effectuer ces actions simultanément pour bénéficier d'un processus aussi rapide que possible. Pour ce faire, combinez simplement plusieurs promesses de coroutine à l'aide de la fonction de promesse `all()`.

```
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 = ['amzn-s3-demo-bucket1', 'amzn-s3-demo-bucket2', 'amzn-s3-demo-bucket3'];
$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();
```

Gestion des erreurs dans la AWS SDK pour PHP version 3

Gestion des erreurs synchrones

Si une erreur se produit lors de l'exécution d'une opération, une exception est levée. Aussi, si vous devez gérer des erreurs dans votre code, utilisez des blocs `try`/`catch` autour de vos opérations. Le kit SDK lève des exceptions spécifiques au service en cas d'erreur.

L'exemple suivant repose sur `Aws\S3\S3Client`. Si une erreur se produit, l'exception levée sera du type `Aws\S3\Exception\S3Exception`. Toutes les exceptions spécifiques au service levées par le kit SDK sont dérivées de la classe `Aws\Exception\AwsException`. Cette classe contient des informations utiles sur l'échec, y compris l'ID de la requête, le code d'erreur et le type d'erreur. Notez que pour certains services qui prennent cela en charge, les données de réponse sont converties dans une structure de tableau associatif (similaire à des objets `Aws\Result`), à laquelle il est possible d’accéder comme à un tableau associatif PHP normal. La méthode `toArray()` renvoie tous ces données, si elles existent.

Importations

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

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

Exemple de code

```
// 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' => 'amzn-s3-demo-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());
}
```

Gestion asynchrone des erreurs

Aucune exception n'est levée lors de l'envoi de requêtes asynchrones. Vous devez utiliser la méthode `then()` ou `otherwise()` de la promesse renvoyée pour recevoir le résultat ou l'erreur.

Importations

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

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

Exemple de code

```
//Asynchronous Error Handling
$promise = $s3Client->createBucketAsync(['Bucket' => 'amzn-s3-demo-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);
});
```

Vous pouvez également « débloquer » la promesse et forcer la levée de l'exception.

Importations

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

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

Exemple de code

```
$promise = $s3Client->createBucketAsync(['Bucket' => 'amzn-s3-demo-bucket']);
```

```
//throw exception
try {
    $result = $promise->wait();
```

```
} catch (S3Exception $e) {
    echo $e->getMessage();
}
```

Gestionnaires et intergiciels dans la version 3 AWS SDK pour PHP

Le principal mécanisme d'extension passe par les AWS SDK pour PHP gestionnaires et les intergiciels. Chaque classe de client de kit SDK possède une instance `Aws\HandlerList` accessible via la méthode `getHandlerList()` d'un client. Vous pouvez récupérer la `HandlerList` d'un client et la modifier afin d'ajouter ou de supprimer un comportement client.

Gestionnaires

Un gestionnaire est une fonction qui transforme une commande et une requête en un résultat. Il envoie généralement des requêtes HTTP. Les gestionnaires peuvent comporter des intergiciels, qui renforcent leur comportement. Un gestionnaire est une fonction qui accepte un `Aws\CommandInterface` et un `Psr\Http\Message\RequestInterface`, et renvoie une promesse exécutée avec un `Aws\ResultInterface` ou rejetée avec une raison `Aws\Exception\AwsException`.

Voici un gestionnaire qui renvoie le même résultat fictif pour chaque appel.

```
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);
};
```

Vous pouvez utiliser ce gestionnaire avec un client de kit SDK en spécifiant une option `handler` dans le constructeur du client.

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

```
]);
```

Vous pouvez également modifier le gestionnaire existant d'un client à l'aide de la méthode `setHandler` de `Aws\ClientInterface`.

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

> **Note**
>
> Pour modifier le gestionnaire d'un client multirégional une fois celui-ci construit, utilisez la `useCustomHandler` méthode d'un. `Aws\MultiRegionClient`
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

Gestionnaire fictif

Nous vous recommandons d'utiliser `MockHandler` lorsque vous écrivez des tests qui utilisent le kit SDK. `Aws\MockHandler` vous permet de renvoyer des résultats fictifs ou de lancer des exceptions de simulation. Vous mettez en file d'attente les résultats ou les exceptions, puis vous les MockHandler replacez dans l'ordre FIFO.

```
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();
```

Intergiciel

Un intergiciel est un type spécial de fonction de haut niveau qui augmente le comportement de transfert d'une commande et délègue la tâche à un gestionnaire « suivant ». Les fonctions d'intergiciel acceptent un `Aws\CommandInterface` et un `Psr\Http\Message\RequestInterface`, et renvoient une promesse exécutée avec un `Aws\ResultInterface` ou rejetée avec une raison `Aws\Exception\AwsException`.

Un intergiciel est une fonction supérieure qui modifie les commandes, les requêtes ou les résultats qu'il traite. Un intergiciel se présente sous la forme suivante.

```
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;
        };
    };
};
```

Un intergiciel reçoit une commande à exécuter et un objet de requête facultatif. Il peut choisir d'augmenter la requête et la commande ou de les laisser en l'état. Un intergiciel appelle le gestionnaire suivant de la chaîne ou choisit de le court-circuiter et de renvoyer une promesse. La promesse ainsi créée en appelant le gestionnaire suivant peut alors être augmentée à l'aide de la méthode `then` de la promesse afin de modifier le résultat ou l'erreur avant le renvoi de la promesse à la pile d'intergiciels.

HandlerList

Le kit SDK utilise `Aws\HandlerList` pour gérer les intergiciels et les gestionnaires utilisés lors de l'exécution d'une commande. Chaque client du kit SDK possède une `HandlerList`. Cette `HandlerList` est clonée et ajoutée à chaque commande créée par un client. Vous pouvez attacher un intergiciel et un gestionnaire par défaut à utiliser pour chaque commande créée par un client en ajoutant un intergiciel à la `HandlerList` du client. Vous pouvez ajouter et supprimer des intergiciels de certaines commandes spécifiques en modifiant la `HandlerList` appartenant à une commande spécifique.

`HandlerList` représente une pile d'intergiciels utilisés pour encapsuler un gestionnaire. Pour vous aider à gérer la liste d'intergiciels et l'ordre dans lequel ils encapsulent un gestionnaire, `HandlerList` divise la pile d'intergiciels en plusieurs étapes nommées qui représentent une partie du cycle de vie du transfert d'une commande :

1. `init` - Ajout des paramètres par défaut
2. `validate` - Validation des paramètres obligatoires
3. `build` - Sérialisation d'une requête HTTP pour envoi
4. `sign` - Signature de la requête HTTP sérialisée
5. **<**handler**>** (il ne s'agit pas ici d'une étape, mais du transfert lui-même)

init

> Cette étape du cycle de vie représente l'initialisation d'une commande. La requête n'a pas encore été sérialisée. Cette étape est généralement utilisée pour ajouter des paramètres par défaut à une commande.
>
> Vous pouvez ajouter un intergiciel à l'étape `init` à l'aide des méthodes `appendInit` et `prependInit`, où `appendInit` ajoute l'intergiciel à la fin de la liste `prepend`, et où `prependInit` ajoute l'intergiciel au début de la liste `prepend`.

```
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');
```

valider

Cette étape du cycle de vie permet de valider les paramètres d'entrée d'une commande.

Vous pouvez ajouter un intergiciel à l'étape `validate` à l'aide des méthodes `appendValidate` et `prependValidate`, où `appendValidate` ajoute l'intergiciel à la fin de la liste `validate`, et où `prependValidate` ajoute l'intergiciel au début de la liste `validate`.

```
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');
```

build

Cette étape du cycle de vie permet de sérialiser une requête HTTP pour la commande en cours d'exécution. Les événements du cycle de vie en aval recevront une commande et une requête HTTP PSR-7.

Vous pouvez ajouter un intergiciel à l'étape `build` à l'aide des méthodes `appendBuild` et `prependBuild`, où `appendBuild` ajoute l'intergiciel à la fin de la liste `build`, et où `prependBuild` ajoute l'intergiciel au début de la liste `build`.

```
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

Cette étape du cycle de vie est généralement utilisée pour signer les requêtes HTTP avant leur envoi sur le réseau. De manière générale, nous vous recommandons d'éviter de muter une requête HTTP qui a déjà été signée afin d'éviter les erreurs de signature.

Il s'agit de la dernière étape de `HandlerList` avant le transfert de la requête HTTP par un gestionnaire.

Vous pouvez ajouter un intergiciel à l'étape `sign` à l'aide des méthodes `appendSign` et `prependSign`, où `appendSign` ajoute l'intergiciel à la fin de la liste `sign`, et où `prependSign` ajoute l'intergiciel au début de la liste `sign`.

```
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');
```

Intergiciels disponibles

Le kit SDK comporte plusieurs intergiciels que vous pouvez utiliser pour augmenter le comportement d'un client ou pour observer l'exécution d'une commande.

mapCommand

L'intergiciel `Aws\Middleware::mapCommand` vous permet de modifier une commande avant qu'elle ne soit sérialisée en tant que requête HTTP. Par exemple, vous pouvez utiliser `mapCommand` pour effectuer une validation ou pour ajouter des paramètres par défaut. La fonction `mapCommand` accepte une fonction de type callable, qui accepte un objet `Aws\CommandInterface` et renvoie un objet `Aws\CommandInterface`.

```
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

L'intergiciel `Aws\Middleware::mapRequest` vous permet de modifier une requête après sa sérialisation, mais avant son envoi. Vous pouvez par exemple l'utiliser pour ajouter des en-têtes HTTP personnalisés à une requête. La fonction `mapRequest` accepte une fonction de type callable, qui accepte un argument `Psr\Http\Message\RequestInterface` et renvoie un objet `Psr\Http\Message\RequestInterface`.

```
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'
);
```

Désormais, lorsque la commande est exécutée, elle est envoyée avec l'en-tête personnalisé.

> **Important**
>
> Notez que l'intergiciel a été ajouté à la liste des gestionnaires à la fin de l'étape `build`. Cela permet de s'assurer qu'une requête a été créée avant l'appel de cet intergiciel.

mapResult

L'intergiciel `Aws\Middleware::mapResult` permet de modifier le résultat de l'exécution d'une commande. La fonction `mapResult` accepte une fonction de type callable, qui accepte un argument `Aws\ResultInterface` et renvoie un objet `Aws\ResultInterface`.

```
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;
    })
);
```

Désormais, lorsque la commande est exécutée, le résultat renvoyé contient un attribut `foo`.

historique

L'intergiciel `history` permet de vérifier que le kit SDK a bien exécuté les commandes attendues, qu'il a bien envoyé les requêtes HTTP spécifiées, et qu'il a bien reçu les résultats escomptés. Globalement, il agit de manière similaire à l'historique d'un navigateur web.

```
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));
```

Un conteneur d'historique `Aws\History` stocke 10 entrées par défaut avant de purger ses entrées. Vous pouvez personnaliser le nombre d'entrées en indiquant combien le constructeur peut en recevoir.

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

Vous pouvez inspecter le conteneur d'historique après l'exécution de requêtes traitées par l'intergiciel d'historique.

```
// 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

L'intergiciel `tap` fonctionne comme un observateur. Vous pouvez l'utiliser pour appeler des fonctions lors de l'envoi de commandes via la chaîne d'intergiciels. La fonction `tap` accepte une fonction de type callable qui accepte `Aws\CommandInterface` et une option `Psr\Http\Message\RequestInterface` en cours d'exécution (facultatif).

```
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";
        }
```

```
    }
);
```

Création de gestionnaires personnalisés

Un gestionnaire est une fonction qui accepte des objets `Aws\CommandInterface` et `Psr\Http\Message\RequestInterface`, et renvoie une promesse `GuzzleHttp\Promise\PromiseInterface` exécutée avec `Aws\ResultInterface` ou rejetée avec `Aws\Exception\AwsException`.

Même si le kit SDK possède plusieurs options `@http`, un gestionnaire n'a besoin de savoir utiliser que les options suivantes :

- connect_timeout
- debug
- decode_content (facultatif)
- delay
- progress (facultatif)
- proxy
- sink
- synchronous (facultatif)
- stream (facultatif)
- timeout
- verify
- http_stats_receiver (facultatif) - Fonction à appeler avec un tableau associatif de statistiques de transfert HTTP sur demande à l'aide du paramètre de configuration stats.

Un gestionnaire DOIT pouvoir gérer l'option ou DOIT renvoyer une promesse rejetée, sauf si l'option est spécifiée comme étant facultative.

En plus de gérer des `@http` options spécifiques, un gestionnaire DOIT ajouter un `User-Agent` en-tête qui prend la forme suivante, où « 3.X » peut être remplacé par `Aws\Sdk::VERSION` et « HandlerSpecificData /version... » doit être remplacé par la chaîne User-Agent spécifique à votre gestionnaire.

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

Streams dans la AWS SDK pour PHP version 3

Dans le cadre de son intégration du standard de message HTTP PSR-7, il AWS SDK pour PHP utilise le PSR-7 StreamInterface en interne comme abstraction sur les flux PHP. Toute commande dont le champ de saisie est défini comme un blob, tel que le `Body` paramètre d'une PutObject commande S3 : :, peut être satisfaite par une chaîne, une ressource de flux PHP ou une instance de`Psr\Http\Message\StreamInterface`.

> **Warning**
>
> Le kit SDK s'approprie n'importe quelle ressource de flux PHP brute fournie en tant que paramètre d'entrée à une commande. Le flux est consommé et fermé en votre nom.
> Si vous avez besoin de partager un flux entre une opération SDK et votre code, encapsulez-le dans une instance `GuzzleHttp\Psr7\Stream` avant de l'inclure en tant que paramètre.
> Le kit SDK consomme le flux ; votre code a donc besoin de compter les mouvements du curseur interne du flux. Les flux Guzzle appellent `fclose` sur la ressource de flux sous-jacente lorsqu'ils sont détruits par le nettoyage de mémoire de PHP. Par conséquent, vous n'avez pas besoin de fermer le flux vous-même.

Décorateurs de stream

Guzzle fournit plusieurs décorateurs de flux que vous pouvez utiliser pour contrôler la façon dont le kit SDK et Guzzle interagissent avec la ressource de streaming fournie en tant que paramètre d'entrée à une commande. Ces décorateurs peuvent modifier la façon dont les gestionnaires sont en mesure de lire et de rechercher sur un flux donné. Voici une liste partielle ; vous pouvez en trouver d'autres dans le référentiel GuzzleHttpPsr 7.

AppendStream

GuzzleHttp\ Page 7 \ AppendStream

Lectures à partir de plusieurs flux, l'un après l'autre.

```
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\ Page 7 \ CachingStream

Utilisé pour autoriser la recherche sur les octets précédemment lus ou sur les flux qui ne peuvent pas être recherchés. Cela peut être utile lorsque le transfert d'un corps d'entité ne pouvant être recherché échoue en raison du besoin de revenir en arrière dans le flux (par exemple, suite à une redirection). Les données qui sont lues à partir du flux distant sont mises en mémoire tampon dans un flux temporaire PHP afin de mettre en cache les octets précédemment lus dans la mémoire, puis sur le disque.

```
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\ Page 7 \ InflateStream

Utilise le filtre PHP zlib.inflate pour gonfler ou dégonfler du contenu compressé avec gzip.

Ce décorateur de flux ignore les 10 premiers octets du flux donné pour supprimer l'en-tête gzip, convertit le flux fourni en ressource de flux PHP, et ajoute ensuite le filtre zlib.inflate. Le flux est ensuite converti en ressource de flux Guzzle pour être utilisé comme flux Guzzle.

LazyOpenStream

GuzzleHttp\ Page 7 \ LazyOpenStream

Lit ou écrit lentement dans un fichier qui est ouvert uniquement après qu'une opération d'I/O est effectuée dans le flux.

```
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\ Page 7 \ LimitStream

Permet de lire un sous-ensemble ou une tranche d'un objet de flux existant. Cela peut être utile pour diviser un gros fichier en petits morceaux à envoyer en plusieurs morceaux (par exemple, l'API Amazon S3 Multipart Upload).

```
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\ Page 7 \ NoSeekStream

Encapsule un flux et n'autorise pas la recherche.

```
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\ Page 7 \ PumpStream

Fournit un flux en lecture seule qui pompe des données depuis une fonction PHP callable.

Lors de l'invocation de l'appelable fourni, il PumpStream transmet la quantité de données à lire à l'appelable. Le callable peut choisir de ne pas tenir compte de cette valeur et de renvoyer plus ou moins d'octets que demandé. Toutes les données supplémentaires renvoyées par l'appelable fourni sont mises en mémoire tampon en interne jusqu'à ce qu'elles soient épuisées à l'aide de la fonction read () du. PumpStream Le callable fourni DOIT renvoyer faux lorsqu'il n'y a pas d'autres données à lire.

Implémentation de décorateurs de flux

La création d'un décorateur de flux est très simple grâce au GuzzleHttp\ Psr7 \. StreamDecoratorTrait Cette caractéristique fournit des méthodes qui implémentent `Psr\Http\Message\StreamInterface` en redirigeant vers un flux sous-jacent. `use` simplement le `StreamDecoratorTrait` et implémentez vos méthodes personnalisées.

Par exemple, supposons que nous voulions appeler une fonction particulière chaque fois que le dernier octet d'un flux est lu. Ceci peut être implémenté en remplaçant la méthode `read()`.

```
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;
    }
}
```

Ce décorateur pourrait être ajouté à n'importe quel flux existant et utilisé comme ceci.

```
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!"
```

Utilisation de résultats paginés dans la AWS SDK pour PHP version 3

Certaines opérations AWS de service sont paginées et donnent des résultats tronqués. Par exemple, l'`ListObjects`opération Amazon S3 ne renvoie que 1 000 objets à la fois. Ces opérations (qui

comportent généralement le préfixe « list » ou « describe ») requièrent l'exécution des demandes suivantes avec des paramètres jeton (ou marqueur) afin de récupérer l'ensemble des résultats.

Les paginateurs sont une fonctionnalité du processus AWS SDK pour PHP qui agit comme une abstraction de ce processus afin de faciliter l'utilisation de la pagination par les développeurs. APIs Un programme de pagination est principalement un itérateur de résultats. Ils est créé grâce à la méthode `getPaginator()` du client. Lorsque vous appelez `getPaginator()`, vous devez fournir le nom de l'opération et ses arguments (comme lors de l'exécution d'une opération). Vous pouvez itérer sur un objet de programme de pagination en utilisant `foreach` pour obtenir les objets `Aws\Result` individuels.

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

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

Objets du paginateur

L'objet renvoyé par cette méthode `getPaginator()` est une instance de la classe `Aws\ResultPaginator`. Cette classe met en œuvre une interface native PHP `iterator` ; c'est pourquoi elle fonctionne avec `foreach`. Il peut également être utilisé avec des fonctions d'itérateur, comme `iterator_to_array`, et s'intègre parfaitement avec les itérateurs SPL comme l'objet `LimitIterator`.

Les objets de programme de pagination ne peuvent contenir qu'une seule « page » de résultats à la fois et sont exécutés lentement. Ceci signifie qu'ils réalisent uniquement les demandes dont ils ont besoin pour obtenir la page de résultats. Par exemple, l'`ListObjects`opération Amazon S3 ne renvoie que 1 000 objets à la fois. Ainsi, si votre compartiment contient environ 10 000 objets, le paginateur devra effectuer 10 requêtes au total. Lorsque vous parcourez les résultats, la première demande est exécutée lorsque vous démarrez l'itération, la deuxième lors de la deuxième itération de la boucle, et ainsi de suite.

Énumération des données à partir des résultats

Les objets de programme de pagination possèdent une méthode appelée `search()`, ce qui vous permet de créer des itérateurs pour les données dans un ensemble de résultats. Lorsque vous appelez`search()`, fournissez une JMESPath expression pour spécifier les données à extraire. Appeler `search()` renvoie un itérateur qui génère les résultats de l'expression sur chaque page de résultats. Ceci est évalué lentement, à mesure que vous parcourez l'itérateur renvoyé.

L'exemple suivant est équivalent à l'exemple de code précédent, mais utilise la méthode `ResultPaginator::search()` pour être plus concis.

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

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

JMESPath les expressions vous permettent de faire des choses assez complexes. Par exemple, si vous souhaitez imprimer toutes les clés d'objet et les préfixes communs (par ex., faire un `ls` d'un compartiment), vous pouvez effectuer les opérations suivantes.

```
// 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";
}
```

Pagination asynchrone

Vous pouvez itérer sur les résultats d'une programme de pagination de manière asynchrone en fournissant un rappel pour la méthode `each()` d'une `Aws\ResultPaginator`. La méthode de rappel est appelée pour chaque valeur générée par le programme de pagination.

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'amzn-s3-demo-bucket'
]);

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

> **Note**
>
> L'utilisation de la méthode `each()` vous permet de paginer sur les résultats d'une opération d'API tout en envoyant d'autres demandes de manière asynchrone.

Une valeur de retour non nulle provenant du rappel sera générée par la promesse basée sur la coroutine sous-jacente. Cela signifie que vous pouvez renvoyer les promesses provenant du rappel qui doivent être résolues avant de continuer l'itération sur les éléments restants, en fusionnant principalement d'autres promesses à l'itération. La dernière valeur non nulle renvoyée par le rappel est le résultat qui répond à la promesse de toutes les promesses en aval. Si la dernière valeur renvoyée est une promesse, la résolution de cette promesse est le résultat qui répond ou rejette les promesses en aval.

```
// 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();
```

Serveurs dans la AWS SDK pour PHP version 3

Les programmes d'attente facilitent les opérations avec les systèmes basés sur la cohérence à terme. Ils offrent une méthode abstraite qui permet de patienter jusqu'à ce qu'une ressource entre dans un état particulier en interrogeant la ressource. Vous pouvez trouver une liste des serveurs pris en charge par un client en consultant la documentation de l'API pour une version unique d'un client de service. Pour y accéder, rendez-vous sur la page du client dans la documentation de l'API, accédez au numéro de version spécifique (représenté par une date) et faites défiler la page vers le bas jusqu'à la section « Serveurs ». Ce lien vous amènera à la section des serveurs de S3.

Dans l'exemple suivant, le client Amazon S3 est utilisé pour créer un compartiment. Le serveur est ensuite utilisé pour patienter jusqu'à ce que le compartiment existe.

```
// 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']);
```

Si le programme d'attente dépasse un certain nombre d'interrogations du compartiment, il lève une exception `\RuntimeException`.

Configuration du serveur

Les programmes d'attente sont gérés par un tableau associatif d'options de configuration. Toutes les options utilisées par un programme d'attente spécifique possèdent des valeurs par défaut. Ces valeurs peuvent toutefois être remplacées pour prendre en charge d'autres stratégies d'attente.

Vous pouvez modifier les options de configuration du programme d'attente en transmettant un tableau associatif d'options `@waiter` à l'argument `$args` des méthodes `waitUntil()` et `getWaiter()` d'un client.

```
// Providing custom waiter configuration options to a waiter
$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'amzn-s3-demo-bucket',
```

```
    '@waiter' => [
        'delay'       => 3,
        'maxAttempts' => 10
    ]
]);
```

delay (int)

Durée d'attente entre deux tentatives d'interrogation (en secondes). Chaque programme d'attente possède une valeur de configuration `delay` par défaut, que vous devrez peut-être modifier selon vos cas d'utilisation spécifiques.

maxAttempts (int)

Nombre maximal de tentatives d'interrogation à émettre avant l'échec du programme d'attente. Cette option vous permet de ne pas attendre une ressource indéfiniment. Chaque programme d'attente possède une valeur de configuration `maxAttempts` par défaut, que vous devrez peut-être modifier selon vos cas d'utilisation spécifiques.

initDelay (int)

Durée d'attente avant la première tentative d'interrogation (en secondes). Cette valeur peut s'avérer utile si vous savez que la ressource que vous attendez mettra un certain temps à atteindre l'état souhaité.

before (callable)

Fonction PHP de type callable appelée avant chaque tentative. La fonction est appelée à l'aide de la commande `Aws\CommandInterface` sur le point d'être exécutée, selon le nombre de tentatives qui ont déjà été exécutées. La fonction de type callable `before` permet de modifier des commandes avant leur exécution ou de fournir des informations d'avancement.

```
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
            );
```

```
        }
    ]
]);
```

En attente asynchrone

En plus de l'attente synchrone, vous pouvez demander à un programme d'attente d'effectuer une attente asynchrone lors de l'envoi d'autres requêtes ou en cas d'attente de plusieurs ressources en même temps.

Vous pouvez accéder à une promesse de programme d'attente en récupérant un programme d'attente à partir d'un client à l'aide de la méthode `getWaiter($name, array $args = [])` du client. Utilisez la méthode `promise()` d'un programme d'attente pour démarrer ce dernier. Une promesse de programme d'attente est exécutée avec la dernière occurrence de `Aws\CommandInterface` exécutée dans le programme d'attente, ou rejetée avec `RuntimeException` en cas d'erreur.

```
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();
```

L'exposition d'une API de programme d'attente basée sur des promesses permet de mettre en œuvre des cas d'utilisation puissants avec une surcharge relativement faible. Imaginons par exemple que vous souhaitiez attendre plusieurs ressources et agir avec le premier programme d'attente résolu avec succès.

```
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();
```

JMESPath expressions dans la AWS SDK pour PHP version 3

JMESPathvous permet de spécifier de manière déclarative comment extraire des éléments d'un document JSON. AWS SDK pour PHP Il dépend de jmespath.php pour alimenter certaines des abstractions de haut niveau, comme Paginators dans la AWS SDK pour PHP version 3 et Waiters dans la AWS SDK pour PHP version 3, mais expose JMESPath également la recherche sur et. `Aws\ResultInterface` `Aws\ResultPaginator`

Vous pouvez jouer avec votre navigateur JMESPath en essayant les JMESPath exemples en ligne. Pour en savoir plus sur le langage, y compris les expressions et fonctions disponibles, consultez la JMESPath spécification.

Les AWS CLIsupports JMESPath. Les expressions écrites pour la sortie CLI sont entièrement compatibles avec les expressions écrites pour le kit AWS SDK pour PHP.

Extraction de données à partir de résultats

L'`Aws\ResultInterface`interface dispose d'une `search($expression)` méthode qui extrait les données d'un modèle de résultat sur la base d'une JMESPath expression. L'utilisation d' JMESPath expressions pour interroger les données d'un objet de résultat peut aider à supprimer le code conditionnel standard et à exprimer de manière plus concise les données extraites.

Pour montrer comment cela fonctionne, nous allons commencer par la sortie JSON par défaut ci-dessous, qui décrit deux volumes Amazon Elastic Block Store (Amazon EBS) attachés à des instances Amazon EC2 distinctes.

```
$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"
        }
    }
}
```

Tout d'abord, nous pouvons récupérer uniquement le premier volume de la liste Volumes à l'aide de la commande suivante.

```
$firstVolume = $result->search('Volumes[0]');
```

Nous allons maintenant utiliser l'expression `wildcard-index [*]` pour parcourir l'ensemble de la liste, extraire, puis renommer trois éléments : `VolumeId` est renommé `ID`, `AvailabilityZone` est renommé `AZ` et `Size` reste `Size`. Nous pouvons extraire et renommer ces éléments à l'aide d'une expression `multi-hash` placée après l'expression `wildcard-index`.

```
$data = $result->search('Volumes[*].{ID: VolumeId, AZ: AvailabilityZone, Size: Size}');
```

Nous obtenons ainsi un tableau de données PHP comme suit :

```
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)
  }
}
```

Dans la notation `multi-hash`, vous pouvez également utiliser des clés de chaînes comme `key1.key2[0].key3` pour extraire des éléments profondément imbriqués au sein de la structure. L'exemple suivant illustre ce processus avec la clé `Attachments[0].InstanceId`, dont l'alias est simplement `InstanceId`. (Dans la plupart des cas, JMESPath les expressions ignorent les espaces blancs.)

```
$expr = 'Volumes[*].{ID: VolumeId,
                     InstanceId: Attachments[0].InstanceId,
                     AZ: AvailabilityZone,
                     Size: Size}';

$data = $result->search($expr);
var_dump($data);
```

L'expression précédente générera les données suivantes :

```
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)
  }
}
```

Vous pouvez également filtrer plusieurs éléments avec l'expression `multi-list` :`[key1, key2]`. Cette opération met en forme tous les attributs filtrés dans une seule liste ordonnée par objet, quel que soit le type.

```
$expr = 'Volumes[*].[VolumeId, Attachments[0].InstanceId, AvailabilityZone, Size]';
$data = $result->search($expr);
var_dump($data);
```

Exécuter la recherche précédente génère les données suivantes :

```
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)
  }
}
```

Utilisez une expression `filter` pour filtrer les résultats sur la valeur d'un champ spécifique. L'exemple de requête suivant produit en sortie uniquement des volumes dans la zone de disponibilité `us-west-2a`.

```
$data = $result->search("Volumes[?AvailabilityZone ## 'us-west-2a']");
```

JMESPath prend également en charge les expressions de fonction. Supposons que vous souhaitiez exécuter la même requête que ci-dessus, mais récupérer tous les volumes dans lesquels le volume se trouve dans une AWS région commençant par « us- ». L'expression suivante utilise la fonction `starts_with`, transmettant un littéral de chaîne de `us-`. Le résultat de cette fonction est ensuite comparé à la valeur JSON littérale de `true`, transmettant uniquement les résultats du prédicat de filtre ayant renvoyés `true` via la projection de filtre.

```
$data = $result->search('Volumes[?starts_with(AvailabilityZone, 'us-') ## `true`]');
```

Extraction de données à partir de paginateurs

Comme vous le savez grâce aux paginateurs du guide de la AWS SDK pour PHP version 3, les `Aws\ResultPaginator` objets sont utilisés pour générer des résultats à partir d'une opération d'API paginable. AWS SDK pour PHP Cela vous permet d'extraire et d'itérer sur des données filtrées à partir d'`Aws\ResultPaginator`objets, en implémentant essentiellement une carte plate au-dessus de l'itérateur dans laquelle le résultat d'une JMESPath expression est la fonction cartographique.

Supposons que vous souhaitiez créer un `iterator` générant uniquement des objets de taille supérieure à 1 Mo à partir d'un compartiment. Cette opération peut être réalisée en créant un programme de pagination `ListObjects`, puis en appliquant une fonction `search()` au programme de pagination et en créant un itérateur « flat-mappé » sur les données paginées.

```
$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);
}
```

Appels Services AWS depuis la AWS SDK pour PHP version 3

Les sections suivantes contiennent des exemples, des didacticiels, des tâches et des guides qui vous montrent comment utiliser les AWS SDK pour PHP AWS services.

Rubriques

- Utiliser les fonctionnalités et options de la AWS SDK pour PHP version 3
- Exemples de code avec conseils pour AWS SDK pour PHP

Utiliser les fonctionnalités et options de la AWS SDK pour PHP version 3

La AWS SDK pour PHP version 3 fournit un support pour des fonctionnalités et options supplémentaires avec lesquelles travailler Service AWS APIs. Les sections de cette rubrique couvrent ces options par service.

Rubriques

- Utilisation du gestionnaire de session DynamoDB avec la version 3 AWS SDK pour PHP
- Fonctionnalités et options d'Amazon S3 dans la AWS SDK pour PHP version 3

Utilisation du gestionnaire de session DynamoDB avec la version 3 AWS SDK pour PHP

Le gestionnaire de session DynamoDB est un gestionnaire de session personnalisé pour PHP qui permet aux développeurs d'utiliser Amazon DynamoDB comme magasin de sessions. L'utilisation de DynamoDB pour le stockage de sessions atténue les problèmes liés à la gestion des sessions dans une application Web distribuée en déplaçant les sessions du système de fichiers local vers un emplacement partagé. DynamoDB est rapide, évolutif, facile à configurer et gère automatiquement la réplication de vos données.

Le gestionnaire de session DynamoDB utilise cette `session_set_save_handler()` fonction pour connecter les opérations DynamoDB aux fonctions de session natives de PHP afin de permettre

un véritable remplacement. Cela inclut la prise en charge de fonctions telles que le verrouillage de session et le nettoyage de la mémoire, qui font partie intégrante du gestionnaire de sessions par défaut de PHP.

Pour plus d'informations sur le service DynamoDB, consultez la page d'accueil d'Amazon DynamoDB.

Utilisation de base

Étape 1 : enregistrer le gestionnaire

Tout d'abord, instanciez et enregistrez le gestionnaire de sessions.

```
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();
```

Étape 2. Créez un tableau pour enregistrer vos sessions

Avant de pouvoir utiliser le gestionnaire de session, vous devez créer un tableau dans lequel stocker les sessions. Vous pouvez le faire à l'avance en utilisant la AWS console pour Amazon DynamoDB ou en utilisant le. AWS SDK pour PHP

Lorsque vous créez ce tableau, utilisez « id » pour le nom de la clé primaire. Nous vous recommandons également de configurer un attribut de durée de vie en utilisant l'attribut « expires » pour bénéficier du nettoyage automatique de la mémoire des sessions.

Étape 3. Utilisez les sessions PHP comme vous le feriez normalement

Une fois que le gestionnaire de session est enregistré et que le tableau existe, vous pouvez écrire et lire depuis la session à l'aide de la variable superglobale `$_SESSION`, tout comme vous le feriez normalement avec le gestionnaire de sessions par défaut de PHP. Le gestionnaire de

session DynamoDB encapsule et résume les interactions avec DynamoDB et vous permet d'utiliser simplement les fonctions et l'interface de session natives de PHP.

```
// 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();
```

Configuration

Vous pouvez configurer le comportement du gestionnaire de session à l'aide des options suivantes. Toutes les options sont facultatives, mais veillez à comprendre les valeurs par défaut.

`table_name`

Nom de la table DynamoDB dans laquelle les sessions doivent être stockées. La valeur par défaut est `'sessions'`.

`hash_key`

Nom de la clé de hachage dans le tableau des sessions DynamoDB. La valeur par défaut est `'id'`.

`data_attribute`

Nom de l'attribut dans la table des sessions DynamoDB dans lequel les données de session sont stockées. La valeur par défaut est `'data'`.

`data_attribute_type`

Type de l'attribut dans la table des sessions DynamoDB dans lequel les données de session sont stockées. La valeur par défaut est `'string'`, mais elle peut éventuellement être définie sur `'binary'`.

`session_lifetime`

La durée de vie d'une session inactive avant de devoir être nettoyée de la mémoire. Si elle n'est pas fournie, la valeur réelle de la durée de vie utilisée sera `ini_get('session.gc_maxlifetime')`.

`session_lifetime_attribute`

Nom de l'attribut dans le tableau des sessions DynamoDB dans lequel le délai d'expiration des sessions est enregistré. La valeur par défaut est `'expires'`.

`consistent_read`

Indique si le gestionnaire de session doit utiliser des lectures cohérentes pour l'opération `GetItem`. La valeur par défaut est `true`.

`locking`

Indique s'il faut utiliser le verrouillage de session. La valeur par défaut est `false`.

`batch_config`

Configuration utilisée pour la suppression par lots pendant le nettoyage de la mémoire. Ces options sont transmises directement aux objets WriteRequestBatchDynamoDB. Déclenchez manuellement le nettoyage de la mémoire via `SessionHandler::garbageCollect()`.

`max_lock_wait_time`

Durée maximale (en secondes) pendant laquelle le gestionnaire de session doit attendre l'acquisition d'un verrouillage avant d'abandonner. La valeur par défaut est `10` et est utilisée uniquement avec le verrouillage de session.

`min_lock_retry_microtime`

Durée minimale (en microsecondes) pendant laquelle le gestionnaire de session doit attendre entre deux tentatives d'acquisition d'un verrouillage. La valeur par défaut est `10000` et est utilisée uniquement avec le verrouillage de session.

`max_lock_retry_microtime`

Durée maximale (en microsecondes) pendant laquelle le gestionnaire de session doit attendre entre deux tentatives d'acquisition d'un verrouillage. La valeur par défaut est `50000` et est utilisée uniquement avec le verrouillage de session.

Pour configurer le Gestionnaire de sessions, spécifiez les options de configuration lorsque vous instanciez le gestionnaire. Le code suivant est un exemple avec toutes les options de configuration spécifiées.

```
$sessionHandler = SessionHandler::fromClient($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,
]);
```

Tarification

Outre les frais de stockage et de transfert de données, les coûts associés à l'utilisation de DynamoDB sont calculés en fonction de la capacité de débit allouée à votre table (voir les détails de tarification d'Amazon DynamoDB). Le débit est mesuré en unités de capacité d'écriture et de capacité de lecture. La page d'accueil d'Amazon DynamoDB indique :

Une unité de capacité de lecture représente une lecture à cohérence forte par seconde (ou deux lectures cohérentes à terme par seconde) pour des éléments dont la taille peut atteindre jusqu'à 4 Ko. Une unité de capacité d'écriture représente une écriture par seconde pour des éléments dont la taille peut atteindre jusqu'à 1 Ko.

Enfin, le débit et les coûts nécessaires pour votre tableau de sessions seront corrélés au trafic attendu et à la taille de session. Le tableau suivant explique le nombre d'opérations de lecture et d'écriture effectuées sur votre table DynamoDB pour chacune des fonctions de session.

Lecture via `session_start()`	• 1 opération de lecture (uniquement 0,5 si `consistent_read` est égal à `false`). • (Conditionnel) 1 opération d'écriture pour supprimer la session si elle a expiré.
Lecture via `session_start()` (Utilisation de verrouillage de session)	• Un minimum de 1 opération d'écriture. • (Conditionnel) Opérations d'écriture supplémentaires pour chaque tentative d'acquérir un verrouillage sur la session. En fonction du temps d'attente entre les verrouill ages et des options de nouvel essai.

	• (Conditionnel) 1 opération d'écriture pour supprimer la session si elle a expiré.
Écriture via `session_write_close()`	• 1 opération d'écriture.
Suppression via `session_destroy()`	• 1 opération d'écriture.
Nettoyage de la mémoire	• 0,5 opération de lecture par tranche de 4 Ko de données du tableau à analyser pour les sessions expirées. • 1 opération d’écriture par élément expiré pour supprimer celui-ci.

Verrouillage de session

Le gestionnaire de session DynamoDB prend en charge le verrouillage de session pessimiste afin d'imiter le comportement du gestionnaire de session par défaut de PHP. Par défaut, cette fonctionnalité est désactivée dans le gestionnaire de session DynamoDB car elle peut entraver les performances et augmenter les coûts, en particulier lorsqu'une application accède à la session à l'aide de requêtes Ajax ou d'iframes. Demandez-vous bien si votre application nécessite le verrouillage de session avant l'activation de celle-ci.

Pour activer le verrouillage de session, paramétrez l'option `'locking'` sur `true` lorsque vous instanciez `SessionHandler`.

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name' => 'sessions',
    'locking'    => true,
]);
```

Collecte des ordures

Configurez un attribut de la TTL dans votre tableau DynamoDB, à l'aide de l'attribut « expires ». Ainsi, la mémoire de vos sessions sera automatiquement nettoyée et vous n'aurez pas à le faire vous-même.

Le gestionnaire de session DynamoDB prend également en charge la collecte des déchets de session à l'aide d'une série d'opérations et. `Scan` `BatchWriteItem` En raison de la nature du

fonctionnement de l'opération `Scan`, et afin de trouver et supprimer toutes les sessions expirées, le processus de nettoyage de la mémoire peut exiger un débit alloué élevé.

Pour cette raison, nous ne prenons pas en charge le nettoyage de la mémoire automatisé. Une meilleure pratique consiste à planifier le nettoyage de la mémoire pendant les heures creuses, lorsqu'un pic de débit consommé ne perturbera pas le reste de l'application. Par exemple, vous pouvez avoir une tâche cron nocturne déclenchant un script pour exécuter le nettoyage de la mémoire. Ce script doit faire quelque chose de similaire à ce qui suit.

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
    'batch_config' => [
        'batch_size' => 25,
        'before' => function ($command) {
            echo "About to delete a batch of expired sessions.\n";
        }
    ]
]);

$sessionHandler->garbageCollect();
```

Vous pouvez également utiliser l'option `'before'` au sein de `'batch_config'` pour introduire des délais sur les opérations `BatchWriteItem` qui sont effectuées par le processus de nettoyage de la mémoire. Cela augmentera le temps nécessaire à la collecte des déchets, mais cela peut vous aider à répartir les demandes effectuées par le gestionnaire de session DynamoDB afin de vous aider à rester proche ou dans les limites de votre capacité de débit allouée pendant la collecte des déchets.

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [
    'table_name'   => 'sessions',
    'batch_config' => [
        'before' => function ($command) {
            $command['@http']['delay'] = 5000;
        }
    ]
]);

$sessionHandler->garbageCollect();
```

Bonnes pratiques

1. Créez votre tableau de sessions dans AWS la région la plus proche géographiquement de vos serveurs d'applications ou dans la même région que celles-ci. Cela garantit la latence la plus faible entre votre application et la base de données DynamoDB.
2. Choisissez avec soin la capacité de débit alloué de votre tableau de sessions. Prenez en compte le trafic attendu pour votre application et la taille attendue de vos sessions. Vous pouvez également utiliser le mode de Read/Write capacité « À la demande » pour votre table.
3. Surveillez le débit consommé via la console AWS de gestion ou Amazon CloudWatch, et ajustez vos paramètres de débit selon les besoins pour répondre aux exigences de votre application.
4. Maintenez des sessions de petite taille (idéalement inférieures à 1 Ko). Les sessions de petite taille sont plus performantes et nécessitent moins de capacité de débit alloué.
5. N'utilisez pas le verrouillage de session, sauf si votre application le nécessite.
6. Au lieu d'utiliser les déclencheurs de nettoyage de la mémoire de session intégrés de PHP, programmez votre nettoyage de la mémoire via une tâche cron ou autre mécanisme de planification, pour qu'elle s'exécute pendant les heures creuses. Utilisez l'option `'batch_config'` à votre avantage.

Autorisations IAM requises

Pour utiliser SessionHhandler DynamoDB, vos informations d'identification configurées doivent être autorisées à utiliser la table DynamoDB que vous avez créée à l'étape précédente. La politique IAM suivante contient les autorisations minimales dont vous avez besoin. Pour appliquer cette politique, remplacez la valeur de la ressource par le nom de ressource Amazon (ARN) de la table que vous avez créée précédemment. Pour plus d'informations sur la création et l'attachement de politiques IAM, consultez la section Gestion des politiques IAM dans le Guide de l'utilisateur IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SessionHandler",
            "Action": [
                "dynamodb:GetItem",
```

```
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:Scan",
                "dynamodb:BatchWriteItem"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-
name"
        }
    ]
}
```

Fonctionnalités et options d'Amazon S3 dans la AWS SDK pour PHP version 3

Cette rubrique décrit les fonctionnalités et options supplémentaires fournies par AWS SDK pour PHP la version 3 pour fonctionner avec Amazon S3.

Rubriques

- Client multirégional Amazon S3 dans la AWS SDK pour PHP version 3
- Enveloppeur de flux Amazon S3 dans la AWS SDK pour PHP version 3
- Transférer des fichiers et des répertoires
- Chiffrement côté client Amazon S3 dans la AWS SDK pour PHP version 3
- Protection de l'intégrité des données avec des checksums

Client multirégional Amazon S3 dans la AWS SDK pour PHP version 3

La AWS SDK pour PHP version 3 fournit un client multirégional générique qui peut être utilisé avec n'importe quel service. Cela permet aux utilisateurs de spécifier à quelle AWS région envoyer une commande en fournissant un paramètre `@region` d'entrée à n'importe quelle commande. En outre, le SDK fournit un client multirégional pour Amazon S3 qui répond intelligemment aux erreurs spécifiques d'Amazon S3 et redirige les commandes en conséquence. Cela permet aux utilisateurs d'utiliser le même client pour parler à plusieurs régions. Cette fonctionnalité est particulièrement utile pour les utilisateurs de l'Amazon S3 Stream Wrapper AWS SDK pour PHP version 3, dont les compartiments résident dans plusieurs régions.

Utilisation de base

Le modèle d'utilisation de base d'un client Amazon S3 est le même, qu'il s'agisse d'un client S3 standard ou de son équivalent multirégional. La seule différence d'utilisation au niveau de la commande est qu'une AWS région peut être spécifiée à l'aide du paramètre `@region` d'entrée.

```
// 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**
>
> Lorsque vous utilisez le client Amazon S3 multirégional, vous ne rencontrerez aucune exception de redirection permanente. Un client Amazon S3 standard lance une instance `Aws\S3\Exception\PermanentRedirectException` lorsqu'une commande est envoyée à la mauvaise région. Un client sur plusieurs régions, quant à lui, réexpédie la commande vers la région appropriée.

Cache de la région du compartiment

Les clients multirégionaux Amazon S3 conservent un cache interne des AWS régions dans lesquelles résident des compartiments donnés. Par défaut, chaque client possède son propre cache en

mémoire. Pour partager un cache entre plusieurs clients ou processus, fournissez une instance de `Aws\CacheInterface` comme option `bucket_region_cache` à votre client sur plusieurs régions.

```
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),
    ],
]);
```

Enveloppeur de flux Amazon S3 dans la AWS SDK pour PHP version 3

Le wrapper de flux Amazon S3 vous permet de stocker et de récupérer des données depuis Amazon S3 à l'aide de fonctions PHP intégrées`file_get_contents`, telles que `fopen`,`copy`,`rename`,, `unlink`,`mkdir`, et`rmdir`.

Vous devez enregistrer le wrapper de flux Amazon S3 pour l'utiliser.

```
$client = new Aws\S3\S3Client([/** options **/]);

// Register the stream wrapper from an S3Client object
$client->registerStreamWrapper();
```

Cela vous permet d'accéder aux compartiments et aux objets stockés dans Amazon S3 à l'aide du `s3://` protocole. L'encapsuleur de flux Amazon S3 accepte les chaînes contenant un nom de compartiment suivi d'une barre oblique et d'une clé d'objet ou d'un préfixe facultatif :. `s3://<bucket>[/<key-or-prefix>]`

> **Note**
>
> L'encapsuleur de flux est conçu pour fonctionner avec les objets et les compartiments sur lesquels vous avez au moins une autorisation de lecture. Cela signifie que votre utilisateur doit avoir l'autorisation d'exécuter `ListBucket` sur les compartiments et `GetObject` sur les objets avec lesquels l'utilisateur doit interagir. Pour les cas d'utilisation où vous ne

disposez pas de ce niveau d'autorisation, nous vous recommandons d'utiliser directement les opérations client Amazon S3.

Télécharger les données

Vous pouvez saisir le contenu d'un objet en utilisant `file_get_contents`. Cependant, soyez prudent avec cette fonction, car elle charge tout le contenu de l'objet dans la mémoire.

```
// Download the body of the "key" object in the "bucket" bucket
$data = file_get_contents('s3://bucket/key');
```

`fopen()`À utiliser lorsque vous travaillez avec des fichiers plus volumineux ou si vous devez diffuser des données depuis Amazon S3.

```
// 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**
>
> Des erreurs d'écriture du fichier sont renvoyées uniquement lorsqu'un appel à `fflush` est effectué. Ces erreurs ne sont pas renvoyées lorsqu'un appel à `fclose` non vidé est effectué. La valeur de retour pour `fclose` sera de `true` s'il ferme le flux, quel que soit le nombre d'erreurs en réponse à son `fflush` interne. Ces erreurs ne sont pas non plus renvoyées lors de l'appel à `file_put_contents`, à cause de la façon dont PHP l'implémente.

Streams consultables ouverts

Les flux ouverts en mode « r » autorisent uniquement la lecture de données depuis le flux, et ne peuvent pas être recherchés par défaut. Cela permet de télécharger les données depuis Amazon S3

de manière véritablement en streaming, les octets précédemment lus n'ayant pas besoin d'être mis en mémoire tampon. Si vous avez besoin qu'un flux puisse être recherché, vous pouvez transférer `seekable` dans les options de contexte de flux d'une fonction.

```
$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);
}
```

Ouvrir des flux pouvant être recherchés vous permet de rechercher des octets qui ont été lus auparavant. Vous ne pouvez pas passer directement aux octets qui n'ont pas encore été lus depuis le serveur distant. Pour autoriser le rappel de données précédemment lues, les données sont mises en mémoire tampon dans un flux PHP temporaire à l'aide d'un décorateur de flux. Lorsque la quantité de données mises en cache dépasse 2 Mo, les données dans le flux temporaire sont transférées de la mémoire au disque. Gardez cela à l'esprit lorsque vous téléchargez des fichiers volumineux depuis Amazon S3 à l'aide du paramètre de contexte de `seekable` flux.

Charger des données

Vous pouvez télécharger des données sur Amazon S3 à l'aide de`file_put_contents()`.

```
file_put_contents('s3://bucket/key', 'Hello!');
```

Vous pouvez télécharger des fichiers plus volumineux en diffusant les données en streaming à l'aide de `fopen()` et d'un mode d'accès du flux « w », « x » ou « a ». L'encapsuleur de flux Amazon S3 ne prend pas en charge les flux de lecture et d'écriture simultanés (par exemple « r+ », « w+ », etc.). Cela est dû au fait que le protocole HTTP ne permet pas la lecture et l'écriture simultanées.

```
$stream = fopen('s3://bucket/key', 'w');
fwrite($stream, 'Hello!');
fclose($stream);
```

> **Note**
>
> Amazon S3 exige qu'un en-tête Content-Length soit spécifié avant que la charge utile d'une demande ne soit envoyée. Par conséquent, les données à télécharger dans une opération `PutObject` sont mises en mémoire tampon grâce à un flux PHP temporaire jusqu'à ce que le flux soit vidé ou fermé.

> **Note**
>
> Des erreurs d'écriture du fichier sont renvoyées uniquement lorsqu'un appel à `fflush` est effectué. Ces erreurs ne sont pas renvoyées lorsqu'un appel à `fclose` non vidé est effectué. La valeur de retour pour `fclose` sera de `true` s'il ferme le flux, quel que soit le nombre d'erreurs en réponse à son `fflush` interne. Ces erreurs ne sont pas non plus renvoyées lors de l'appel à `file_put_contents`, à cause de la façon dont PHP l'implémente.

modes fopen

La fonction PHP fopen() nécessite que vous indiquiez une option `$mode`. L'option de mode spécifie si les données peuvent être lues ou écrites dans un flux, et si le fichier doit exister lors de l'ouverture d'un flux.

Le wrapper de flux Amazon S3 prend en charge les modes suivants pour les flux qui ciblent des objets Amazon S3.

r	Un flux en lecture seule où l'objet doit déjà exister.
w	Un flux en écriture seule. Si l'objet existe déjà, il est remplacé.
a	Un flux en écriture seule. Si l'objet existe déjà, il est téléchargé dans un flux temporaire et toutes les écritures dans le flux sont ajoutées aux données précédemment téléchargées.

x	Un flux en écriture seule. Une erreur est générée si l'objet existe déjà.

Autres fonctions de l'objet

Les wrappers de flux permettent à de nombreuses fonctions PHP intégrées de fonctionner avec un système personnalisé tel qu'Amazon S3. Voici quelques-unes des fonctions que le wrapper de flux Amazon S3 vous permet d'exécuter avec des objets stockés dans Amazon S3.

unlink()	Pour supprimer un objet dans un compartiment.
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`
	Vous pouvez transmettre toutes les options disponibles pour l'opération `DeleteObject` afin de modifier la façon dont l'objet est supprimé (par exemple, en spécifiant une version d'objet spécifique).
	`// Delete a specific version of an object from a bucket`
	`unlink('s3://bucket/key', stream_context_create([`
	`'s3' => ['VersionId' => '123']`
	`]);`

filesize()	Pour obtenir la taille d'un objet.
	`// Get the Content-Length of an object`
	`$size = filesize('s3://bucket/key', );`

is_file()	Pour vérifier si une URL est un fichier.
	`if (is_file('s3://bucket/key')) {`
	`echo 'It is a file!';`

	`}`

file_exists()	Pour vérifier si un objet existe.
	`if (file_exists('s3://bucket/key'))`
	`{`
	`echo 'It exists!';`
	`}`

filetype()	Vérifie si une URL est mappée à un fichier ou un compartiment (dir).
dans le fichier()	Pour charger le contenu d'un objet dans un tableau de lignes. Vous pouvez transmett re toute option disponible à l'opération `GetObject` pour modifier la façon dont le fichier est téléchargé.
filemtime()	Pour obtenir la date à laquelle l'objet a été modifié pour la dernière fois.
rename()	Pour renommer un objet en le copiant puis en supprimant l'original. Vous pouvez transmett re les options disponibles pour les opération s `CopyObject` et `DeleteObject` aux paramètres de contexte de flux pour modifier la façon dont l'objet est copié et supprimé.

> **Note**
>
> Bien qu'il fonctionne `copy` généralement avec le wrapper de flux Amazon S3, certaines erreurs peuvent ne pas être correctement signalées en raison des éléments internes de la `copy` fonction en PHP. Nous vous recommandons d'utiliser une instance d'AWSs3 à la place ObjectCopier.

Travaillez avec des compartiments et des dossiers

Utilisable `mkdir()` pour travailler avec des seaux

Vous pouvez créer et parcourir des compartiments Amazon S3 de la même manière que PHP vous permet de créer et de parcourir des répertoires sur votre système de fichiers.

Voici un exemple de création d'un bucket.

```
mkdir('s3://amzn-s3-demo-bucket');
```

> **Note**
>
> En avril 2023, Amazon S3 a automatiquement activé S3 Block Public Access et désactivé les listes de contrôle d'accès pour tous les buckets nouvellement créés. Cette modification affecte également le fonctionnement `StreamWrapper` de la `mkdir` fonction avec les autorisations et ACLs. Plus d'informations sont disponibles dans cet AWS article Quoi de neuf ?

Vous pouvez transmettre des options de contexte de flux à la `mkdir()` méthode pour modifier la façon dont le bucket est créé à l'aide des paramètres disponibles pour l'CreateBucketopération.

```
// Create a bucket in the EU (Ireland) Region
mkdir('s3://amzn-s3-demo-bucket', 0500, true,
        stream_context_create([
            's3' => ['LocationConstraint' => 'eu-west-1']
        ]));
```

Vous pouvez supprimer des compartiments à l'aide de la fonction `rmdir()`.

```
// Delete a bucket
rmdir('s3://amzn-s3-demo-bucket);
```

> **Note**
>
> Un compartiment ne peut être supprimé que s'il est vide.

`mkdir()`À utiliser pour travailler avec des dossiers

Après avoir créé un bucket, vous pouvez l'utiliser `mkdir()` pour créer des objets qui fonctionnent comme des dossiers, comme dans un système de fichiers.

L'extrait de code suivant ajoute un objet de dossier nommé « my-folder » au bucket existant nommé « amzn-s3-demo-bucket ». Utilisez la barre oblique (/) pour séparer le nom d'un objet de dossier du nom du compartiment et de tout autre nom de dossier supplémentaire.

```
mkdir('s3://amzn-s3-demo-bucket/my-folder')
```

La remarque précédente concernant les modifications d'autorisation après avril 2023 entre également en ligne de compte lorsque vous créez des objets de dossier. Ce billet de blog contient des informations sur la façon d'ajuster les autorisations si nécessaire.

Utilisez cette `rmdir()` fonction pour supprimer un objet de dossier vide, comme indiqué dans l'extrait de code suivant.

```
rmdir('s3://amzn-s3-demo-bucket/my-folder')
```

Répertorier le contenu d'un bucket

Vous pouvez utiliser les fonctions PHP opendir (), readdir (), rewinddir () et closedir () avec le wrapper de flux Amazon S3 pour parcourir le contenu d'un bucket. Vous pouvez transmettre les paramètres disponibles pour l'ListObjectsopération sous forme d'options contextuelles de flux personnalisées à la `opendir()` fonction afin de modifier la façon dont les objets sont répertoriés.

```
$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);
}
```

Vous pouvez répertorier de manière récursive chaque objet et chaque préfixe d'un bucket à l'aide de PHP. RecursiveDirectoryIterator

```
$dir = 's3://bucket';
```

```
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

Une autre façon de répertorier de façon récursive les contenus d'un compartiment, qui entraîne moins de requêtes HTTP, est d'utiliser la fonction `Aws\recursive_dir_iterator($path, $context = null)`.

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

$iter = Aws\recursive_dir_iterator('s3://bucket/key');
foreach ($iter as $filename) {
    echo $filename . "\n";
}
```

Options de contexte de diffusion

Vous pouvez personnaliser le client utilisé par l'encapsuleur de flux, ou le cache utilisé pour mettre en cache les informations précédemment chargées sur les compartiments et les clés, en transmettant les options de contexte de flux personnalisé.

L'encapsuleur de flux prend en charge les options de contexte de flux suivantes au niveau de chaque opération.

`client`

L'objet `Aws\AwsClientInterface` à utiliser pour exécuter des commandes.

`cache`

Une instance de `Aws\CacheInterface` à utiliser pour mettre en cache les statistiques des fichiers précédemment obtenues. Par défaut, l'encapsuleur de flux utilise un cache LRU en mémoire.

Transférer des fichiers et des répertoires

La AWS SDK pour PHP version 3 propose deux approches pour le transfert de fichiers et de répertoires vers et depuis Amazon S3. Les deux solutions gèrent la complexité des chargements

et téléchargements partitionnés pour des fichiers volumineux, mais elles diffèrent en termes de philosophie de conception, de fonctionnalités et de modèles d'utilisation.

Vue d'ensemble des options de transfert

Choisissez l'approche de transfert la mieux adaptée aux besoins de votre application :

Gestionnaire de transfert S3 (recommandé)

Une bibliothèque moderne de haut niveau qui fournit une solution complète pour les transferts de fichiers. Il offre de nombreuses options de configuration, un suivi intégré de la progression, des gestionnaires de téléchargement personnalisés et une gestion robuste des erreurs. Le S3 Transfer Manager utilise une API basée sur les promesses et prend en charge à la fois les opérations de fichiers individuels et les transferts de répertoires grâce à des fonctionnalités de filtrage avancées.

Transfert

Une implémentation de transfert d'annuaire qui se concentre spécifiquement sur les opérations d'annuaire en masse. Il fournit une API plus simple pour le chargement et le téléchargement de répertoires entiers avec des options de configuration de base. Cette approche comporte moins de fonctionnalités que le gestionnaire de transfert S3.

Principales différences

Le tableau suivant met en évidence les principales différences entre les deux approches de transfert :

Fonctionnalité	Gestionnaire de transfert S3	Transférer
Opérations sur des fichiers individuels	Oui (télécharger/télécharger des fichiers uniques)	Non (opérations d'annuaire uniquement)
Opérations du répertoire	Oui (avec filtrage avancé)	Oui (transferts de répertoire de base)
Suivi des progrès	Intégré avec des écouteurs personnalisés	Limité (sortie de débogage uniquement)
Gestionnaires de télécharg ement personnalisés	Oui	Non

Fonctionnalité	Gestionnaire de transfert S3	Transférer
Validation du checksum	Automatique avec configuration	Manuel (option add_content_md5)
Gestion des erreurs	Complet avec politiques de défaillance	Gestion de base basée sur les promesses
Options de configuration	Vaste (plus de 8 options)	Basique (6 options)
Conception d'API	Objets de demande/réponse	Paramètres de constructeur simples

Choix de la bonne approche

Utilisez le gestionnaire de transfert S3 lorsque vous avez besoin de :

- Opérations de chargement ou de téléchargement de fichiers individuels
- Suivi et surveillance avancés des progrès
- Gestionnaires de téléchargement personnalisés pour un traitement spécialisé
- Politiques complètes de gestion des erreurs et de nouvelles tentatives
- Contrôle précis des opérations en plusieurs parties
- Opérations d'annuaire avec logique de filtrage complexe

Utilisez Transfer lorsque vous avez besoin de :

- Transferts simples de répertoire vers/depuis-S3
- Configuration et installation minimales
- Compatibilité avec le code existant qui utilise Transfer
- Fonctionnalité de base de téléchargement partitionné

> **Note**
>
> Pour les nouvelles applications, nous recommandons d'utiliser le gestionnaire de transfert S3, car il fournit une solution plus complète et plus flexible pour les transferts de fichiers.

Gestionnaire de transfert S3

Le gestionnaire de transfert S3 fournit une interface pour le chargement et le téléchargement de fichiers vers et depuis Amazon S3. Vous pouvez l'utiliser pour des opérations sur un seul fichier ou des opérations de répertoire.

Le gestionnaire de transfert gère automatiquement les chargements partitionnés et les téléchargements. Il gère les fichiers volumineux et suit les progrès. La bibliothèque met en œuvre les meilleures pratiques pour les transferts de données S3. Vous pouvez l'utiliser pour créer des capacités de transfert de fichiers dans vos applications PHP.

Fonctions principales

Le gestionnaire de transfert S3 fournit les fonctionnalités clés suivantes :

- API simple : chargez et téléchargez des fichiers et des répertoires
- Gestionnaires de téléchargement personnalisés : implémentez votre propre logique de téléchargement
- Chargements et téléchargements partitionnés automatiques : gestion automatique des fichiers volumineux
- Traitement simultané : optimisation du débit
- Suivi de la progression : surveillance de l'état du transfert
- Comportement personnalisable : configurez des options étendues
- Gestion des erreurs : configurez les politiques relatives aux tentatives et aux échecs
- Opérations de répertoire : transfert de plusieurs fichiers par lots
- Validation par somme de contrôle : garantir l'intégrité des données

Installation

Le gestionnaire de transfert S3 est inclus dans la AWS SDK pour PHP version 3. Il n'est pas nécessaire de l'installer séparément.

Pour effectuer l'installation à l'aide de Composer, exécutez la commande suivante :

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

Cette commande installe la AWS SDK pour PHP version 3 complète, y compris le gestionnaire de transfert S3.

Utilisation de base

L'exemple suivant montre comment utiliser le gestionnaire de transfert S3 :

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

// Create a transfer manager with default configuration.
$transferManager = new S3TransferManager($s3Client);

// Alternative: Create transfer manager with null client. S3 Transfer Manager uses a
 default S3 client.
$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Example: Upload a file.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/local/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'path/to/s3/file.txt',
        ]
    )
);

// Wait for the upload to complete.
$result = $uploadPromise->wait();

echo "Upload complete!\n";
```

> **Important**
>
> Lorsque vous utilisez S3 Transfer Manager pour créer un client Amazon S3 par défaut, les clients peuvent transmettre une région par défaut au client en utilisant le `default_region` paramètre des `config` options du gestionnaire de transfert, sinon le client Amazon S3 essaie de résoudre une région en utilisant le comportement par défaut pour résoudre les configurations, et si la région n'est pas résolue, une exception est déclenchée.

Création d'un gestionnaire de transferts

Vous pouvez créer un gestionnaire de transferts de deux manières :

Avec un client S3 existant

Transmettez une `S3Client`instance existante au `S3TransferManager` *<add link>* constructeur.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\S3TransferManager;

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

// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

$transferManager = new S3TransferManager($s3Client);
```

Avec la création d'un client S3 par défaut

Passez `null` en tant que client et spécifiez les options de configuration.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;
```

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);
```

Configuration

S3 Transfer Manager accepte les options de configuration pour personnaliser son comportement. Fournissez ces options lorsque vous créez une instance du gestionnaire de transferts. Le paramètre de configuration peut être un tableau ou une instance de `S3TransferManagerConfig`*<add link>*.

L'exemple suivant configure une instance de S3 Transfer Manager :

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null,
    [
        // 10MB parts for multipart operations.
        'target_part_size_bytes' => 10 * 1024 * 1024,

        // Use multipart upload for files larger than 20MB.
        'multipart_upload_threshold_bytes' => 20 * 1024 * 1024,

        // Enable checksum calculation for data integrity.
        'request_checksum_calculation' => 'when_supported',

        // Enable checksum validation when getting objects.
        'response_checksum_validation' => 'when_supported',

        // Use part-based multipart downloads.
        'multipart_download_type' => 'part',

        // Allow up to 10 concurrent operations.
        'concurrency' => 10,
```

```
        // Enable progress tracking.
        'track_progress' => true,

        // Set default region for default S3 client construction.
        'default_region' => 'us-west-2',
    ]
);
```

> **Note**
>
> Lorsque vous fournissez une configuration sous forme de tableau à`S3TransferManager`, le SDK appelle en interne `S3TransferManagerConfig::fromArray` pour la convertir dans le type approprié.

Options de configuration

Toutes les options de configuration sont facultatives et utilisent des valeurs par défaut lorsqu'elles ne sont pas spécifiées.

Option	Type	Par défaut	Description
`target_part_size_bytes`	int	8 MO	Taille de pièce minimale pour les chargements/téléchargements partitionnés.
`multipart_upload_threshold_bytes`	int	16 Mo	Seuil de taille de fichier pour utiliser le téléchargement en plusieurs parties.
`request_checksum_calculation`	chaîne	« lorsqu'il est pris en charge »	Active le calcul de la somme de contrôle. Les valeurs valides sont « when_supported », « when_required ».
`response_checksum_validation`	chaîne	« lorsqu'il est pris en charge »	Active la validation de la somme de contrôle lors de l'obtention d'objets. Les valeurs valides sont

Option	Type	Par défaut	Description
			« when_supported », « when_requ ired ».
`multipart _download _type`	chaîne	'partie'	Stratégie de téléchargement pour les fichiers volumineux. Les valeurs valides sont « part » (téléchar gement en plusieurs parties), « rangé » (demandes de plage).
`concurrency`	int	5	Nombre maximum d'opérations simultanées.
`track_pro gress`	bool	FALSE	S'il faut suivre la progression du transfert.
`default_r egion`	chaîne	'us-east-1'	Région AWS à utiliser si aucun client S3 n'est fourni.

Opérations sur les fichiers

S3 Transfer Manager fournit des méthodes pour charger et télécharger des fichiers individuels.

Téléchargez des fichiers locaux

Pour charger un fichier sur Amazon S3, utilisez la `upload` *<add link>* méthode.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Source can be from a local path to a file.
$source = '/path/to/local/file.txt';
```

```
// Or the source can be an instance of StreamInterface.
$source = GuzzleHttp\Psr7\Utils::streamFor('Hello World!');

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        $source,
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'path/to/s3/file.txt',
            // Additional `putObject` parameters as needed.
        ],
        [
            // Optional configuration overrides.
            'multipart_upload_threshold_bytes' => 100 * 1024 * 1024, // 100MB
            'target_part_size_bytes'           => 10 * 1024 * 1024, // 10MB
            'track_progress'                   => true,
            'request_checksum_calculation'     => 'when_required',
        ]
    )
);

// The upload is asynchronous, you can wait for it to complete.
$result = $uploadPromise->wait();

// Or you can use the promise for more complex workflows.
$result = $uploadPromise->then(
    function ($result) {
        echo "Upload succeeded!";
    },
    function ($error) {
        echo "Upload failed: " . $error->getMessage();
    }
)->wait();
```

Paramètres de la méthode **`upload`**

La `upload` méthode accepte une instance de `UploadRequest` *<add link>* comme argument.

Paramètres dans l'**`UploadRequest`**

Paramètre	Type	Obligatoire	Description
`$source`	chaîne| `StreamInterface`	Oui	Chemin du chemin du fichier local ou du flux contenant les données à télécharger.
`$uploadRequestArgs`	array	Oui	Téléchargez les arguments de la demande (doivent inclure Bucket et Key).
`$config`	array	Non	Les remplacements de configuration spécifiques à ce téléchargement. Pour plus d'informations sur les options de configuration, consultez la section suivante.
`$listeners`	array	Non	Tableau d'`TransferListener` objets permettant de surveiller ce téléchargement.
`$progressTracker`	TransferListener	Non	Un outil de suivi de la progression de ce téléchargement.

`$config`options pour **`UploadRequest`**

Le SDK résout la `$config` valeur par défaut à partir de la configuration du gestionnaire de transfert S3.

Option	Type	Obligatoire	Description
`multipart_upload_threshold_bytes`	int	Non	Remplace le seuil par défaut lorsqu'un téléchargement partitionné est déclenché.

Option	Type	Obligatoire	Description
`target_part_size_bytes`	int	Non	Remplace la taille de pièce cible par défaut en octets.
`track_progress`	bool	Non	Remplace l'option par défaut pour activer le suivi de la progression. Si cette option existe `true` et que vous ne fournissez aucun `progressTracker` paramètre, le SDK utilise une implémentation par défaut.
`concurrency`	int	Non	Remplace la valeur par défaut pour la simultanéité.
`request_checksum_calculation`	chaîne	Non	Remplace la valeur par défaut indiquant si le calcul de la somme de contrôle de la demande doit être effectué.

Lorsque la `upload` méthode s'exécute correctement, elle renvoie un `UploadResult`*<add link>*.

Télécharger des objets S3

Pour télécharger un objet depuis Amazon S3, utilisez la `download` *<add link>* méthode.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Download using an S3 URI.
$downloadPromise = $transferManager->download(
```

```
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
        [
            // Additional `getObject` parameters as needed.
        ],
        [
            // Optional configuration overrides.
            'response_checksum_validation' => 'when_required',
            'track_progress'               => true,
            'target_part_size_bytes'       => 10 * 1024 * 1024, // 10MB
        ]
    )
);

// Wait for the download to complete.
$result = $downloadPromise->wait();

// The SDK uses a stream-based download handler by default.
$stream = $result->getDownloadDataResult();

// You can either get the content.
$content = $stream->getContents();

// Or write the stream to a file.
file_put_contents('/path/to/local/file.txt', $stream);

// Don't forget to close the stream.
$stream->close();
```

Paramètres de la méthode **`download`**

La `download` méthode accepte une instance de `DownloadRequest` *<add link>* comme argument.

DownloadRequest paramètres

Paramètre	Type	Obligatoire	Description
`$source`	chaîne|tableau|`null`	Non	L'objet à télécharger depuis S3. Vous pouvez fournir ce paramètre sous la forme d'une chaîne d'URI S3 (» s3://amzn-s3-demo-bucket/key «), d'un

Paramètre	Type	Obligatoire	Description
			tableau avec les clés Bucket et Key, ou`null`. Lorsque cette valeur est définie`null`, transmettez les valeurs Bucket et Key dans le `$download RequestArgs` paramètre.
`$download RequestAr gs`	array	Non	Arguments de demande d'objet de téléchargement supplémentaires. Dans `$source` l'affirmative`null`, indiquez les valeurs du bucket et de la clé ici.
`$config`	array	Non	Les remplacements de configuration spécifiques à ce téléchargement. Pour plus d'informations sur les options de configuration, consultez la section suivante.

Paramètre	Type	Obligatoire	Description
`$downloadHandler`	`AbstractDownloadHandler` *<add link>*|`null`	Non	Le gestionnaire qui reçoit chaque fragment d'objet et gère son téléchargement. Le gestionnaire par défaut de la `download` méthode utilise un gestionnaire basé sur les flux. Ce gestionnaire pousse chaque fragment d'objet dans un flux. Le gestionnaire par défaut de la `downloadFile` méthode (voirthe section called “Télécharger un objet S3 dans un fichier local”) utilise un gestionnaire basé sur des fichiers. Ce gestionnaire écrit chaque fragment dans un fichier. Vous pouvez implémenter le vôtre `DownloadHandler` pour personnaliser l'emplacement et la manière dont le SDK stocke les données téléchargées. Par exemple, vous pouvez stocker des données dans des bases de données, un stockage dans le cloud ou des pipelines de traitement personnalisés au lieu de fichiers ou de flux.
`$progressTracker`	`TransferListener`	Non	Un outil de suivi de la progression de ce téléchargement.
`$listeners`	`array`	Non	Tableau d'`AbstractTransferListener` objets permettant de surveiller ce téléchargement.

`$config`options pour **`DownloadRequest`**

Le SDK résout la `$config` valeur par défaut à partir de la configuration du gestionnaire de transfert S3.

Option	Type	Obligatoire	Description
`multipart_download_type`	chaîne	Non	Remplace le type de téléchargement en plusieurs parties par défaut. Les valeurs valides sont « part », « ranged »
`response_checksum_validation`	chaîne	Non	Remplace la valeur résolue dans la configuration du gestionnaire de transfert pour déterminer si la validation de la somme de contrôle doit être effectuée. Le SDK prend en compte cette option uniquement si elle n'`ChecksumMode` est pas présente dans `$getObjectRequestArgs` le `DownloadRequest` .
`track_progress`	bool	Non	Remplace l'option par défaut permettant d'activer le suivi de la progression. Si cette option existe `true` et que vous ne fournissez pas de `progressTracker` paramètre avec `DownloadRequest,` le SDK, il utilise une implémentation par défaut. Utilisez cette option pour activer un outil de suivi de progression par défaut lorsque le `$progressTracker` paramètre est défini sur`null`.

Option	Type	Obligatoire	Description
`target_part_size_bytes`	int	Non	Taille de partie en octets à utiliser dans un téléchargement multipartie par plage. Si vous ne fournissez pas ce paramètre, le téléchargement utilise le paramètre `target_part_size_bytes` configuré dans le gestionnaire de transfert.

Lorsque la `download` méthode s'exécute correctement, elle renvoie un `DownloadResult`*<add link>*.

Télécharger un objet S3 dans un fichier local

Pour télécharger un objet S3 directement dans un fichier local, utilisez la `downloadFile` méthode suivante :

```
<?php

use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$downloadFilePromise = $transferManager->downloadFile(
    new DownloadFileRequest(
        '/path/to/local/file.txt',  // Destination file path.
        false,  // Fail when destination exists.
        new DownloadRequest(
            's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
            [],  // `getObject` request args
            [
                'track_progress'         => true,
                'target_part_size_bytes' => 10 * 1024 * 1024, // 10MB parts
            ]
```

```
        )
    )
);

// Wait for download to complete.
$result = $downloadFilePromise->wait();

// `getDownloadDataResult()` returns the string for the file path of the downloaded
 content because
//  the default `DownloadHandler` used by `downloadFile()` is a file-based handler.
echo "File downloaded successfully at {$result->getDownloadDataResult()}!\n";
```

Paramètres de la méthode **`downloadFile`**

La `downloadFile` méthode accepte une instance de `DownloadFileRequest` comme argument.

Paramètres dans l'**`DownloadFileRequest`**

Option	Type	Obligatoire	Description
`$destination`	chaîne	Oui	Chemin local dans lequel le fichier est enregistré.
`$failsWhenDestinationExists`	bool	Oui	S'il faut échouer si le fichier de destination existe déjà. Si cette option est activée `false` et que le fichier de destination existe, le SDK supprime ou remplace le fichier existant.
`$downloadRequest`	`DownloadRequest`	Oui	L'`DownloadRequest` objet configuré. Pour plus d'informations, voir `DownloadRequest` objet.

Lorsque la `downloadFile` méthode s'exécute correctement, elle renvoie un `DownloadResult`*`<add link>`*.

Opérations du répertoire

S3 Transfer Manager peut gérer des transferts d'annuaires complets.

Répertoire de téléchargement

S3 Transfer Manager peut télécharger un répertoire entier dans un compartiment S3. *<Are the key values the filenames when uploaded?>*

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/local/directory',
        'amzn-s3-demo-bucket',
        [
            // Additional `putObject` parameters that apply to all files.
            'ACL'          => 'public-read',
            'CacheControl' => 'max-age=3600',
        ],
        [
            // Configuration options.
            'recursive'                     => true,
            'follow_symbolic_links'         => false,
            's3_prefix'                     => 'uploads/2023/',
            's3_delimiter'                  => '/',
            'track_progress'                => true,
            'filter'                        => function ($file) {
                // Upload only .jpg files.
                return pathinfo($file, PATHINFO_EXTENSION) === 'jpg';
            },
            'upload_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ContentType'] = 'image/jpeg';
            },
        ]
    )
);
```

```
// Wait for the upload process to complete.
$result = $uploadDirPromise->wait();

$uploaded = $result->getObjectsUploaded();
$failed = $result->getObjectsFailed();
echo "Uploaded {$uploaded} files. Failed: {$failed}\n";
```

Paramètres de la méthode **`uploadDirectory`**

La `uploadDirectory` méthode accepte une instance de `UploadDirectoryRequest` *<add link>* comme argument.

Paramètres dans l'**`UploadDirectoryRequest`**

Paramètre	Type	Obligatoire	Description
`$sourceDirectory`	chaîne	Oui	Chemin du répertoire local à télécharger.
`$targetBucket`	chaîne	Oui	Nom du compartiment S3 de destination.
`$uploadRequestArgs`	array	Non	Paramètres de demande d'objet de téléchargement supplémentaires pour tous les fichiers.
`$config`	array	Non	Options de configuration pour le téléchargement du répertoire. Pour plus d'informations sur les options de configuration, consultez la section suivante.
`$listeners`	array	Non	Tableau d'objets `TransferListener` . Le SDK clone chacun d'eux `TransferListener` par fichier.
`$max_depth`	bool	-1 « Exécution par défaut »	Pour indiquer la profondeur maximale du fichier récursif, trois marches.

Paramètre	Type	Obligatoire	Description
`$progress Tracker`	`TransferL istener`	Non	Un outil de suivi de la progression de tous les téléchargements.

`$config`options pour **`UploadDirectoryRequest`**

Option	Type	Par défaut	Description
`recursive`	bool	`false`	S'il faut télécharger des sous-répe rtoires de manière récursive.
`follow_sy mbolic_li nks`	bool	`false`	S'il faut suivre des liens symboliques.
`s3_prefix`	chaîne	''	Préfixe à ajouter à toutes les clés d'objet.
`s3_delimi ter`	chaîne	'/'	Le délimiteur à utiliser dans les clés d'objet S3.
`track_pro gress`	bool	`false`	S'il faut suivre les progrès.
`max_concu rrency`	int	100	Le nombre maximum de télécharg ements simultanés.
`filter`	appelable	null	Fonction permettant de filtrer les fichiers à télécharger.
`upload_ob ject_requ est_modif ier`	appelable	null	Fonction permettant de personnaliser chaque demande de téléchargement.
`failure_p olicy`	appelable	null	Fonction permettant de gérer les échecs de téléchargement.

Types d'options appelables **`$config`**

Nom d'option	nom du paramètre	type de paramètre	informations sur les paramètres
`filter`	`$file`	`SplFileInfo |` chaîne	S'il est converti en chaîne, il s'agit du chemin du fichier. Dans le cas contraire, il s'agit d'une instance de`SplFileInfo` .
`upload_object_request_modifier`	`$requestArgs`	array	Les arguments de demande pour chaque demande de téléchargement individuelle. Pour plus d'informations sur les options de tableau, consultez la section sur la syntaxe des paramètres de la méthode putObject.
`failure_policy`	`$uploadRequestArgs`	array	Les arguments de demande pour chaque demande de téléchargement individuelle. Pour plus d'informations sur les options de tableau, consultez la section sur la syntaxe des paramètres de la méthode putObject pour les options de tableau.

Nom d'option	nom du paramètre	type de paramètre	informations sur les paramètres
	`$uploadDirectoryArgs`	array	Un tableau contenant le répertoire source et le compartiment cible pour l'opération de répertoire de téléchargement.
	`$reason`	`Throwable` |chaîne	Le support d'exception qui contient des informations sur la cause de l'échec.
	`$uploadDirectoryResult`	`UploadDirectoryResult`	Objet contenant le nombre d'objets chargés avec succès et le nombre d'objets ayant échoué.

Lorsque la `uploadDirectory` méthode s'exécute correctement, elle renvoie un `UploadDirectoryResult`*<add link>*.

Répertoire de téléchargement

Vous pouvez télécharger un répertoire depuis un compartiment S3. *<Are the filenames the key values when downloaded? If there is no extension in the key, is there also no extension on the file?>*

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(null, [
```

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

$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/local/directory',
        [
            // Additional `getObject` parameters that apply to all files.
        ],
        [
            // Configuration options.
            's3_prefix'                       => 'uploads/2023/',
            's3_delimiter'                    => '/',
            'track_progress'                  => true,
            'filter'                          => function ($key) {
                // Download only .jpg files.
                return pathinfo($key, PATHINFO_EXTENSION) === 'jpg';
            },
            'download_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ResponseContentType'] = 'image/jpeg';
            },
            'list_objects_v2_args'             => [
                'MaxKeys' => 1000,
                'Prefix'  => 'uploads/2023/',
            ],
            'fails_when_destination_exists'    => true,
        ]
    )
);

// Wait for the download process to complete.
$result = $downloadDirPromise->wait();

$downloaded = $result->getObjectsDownloaded();
$failed = $result->getObjectsFailed();
echo "Downloaded {$downloaded} files. Failed: {$failed}\n";
```

Paramètres de la méthode **`downloadDirectory`**

La `downloadDirectory` méthode accepte une instance de `DownloadDirectoryRequest` *<add link>* comme argument.

Paramètres dans l'**`DownloadDirectoryRequest`**

Paramètre	Type	Obligatoire	Description
`$sourceBucket`	chaîne	Oui	Nom du compartiment S3 source à partir duquel les objets sont téléchargés.
`$destinationDirectory`	chaîne	Oui	Répertoire local dans lequel télécharger les fichiers.
`$downloadRequestArgs`	array	Non	Paramètres de demande d'objet de téléchargement supplémentaires pour tous les fichiers.
`$config`	array	Non	Options de configuration pour le téléchargement du répertoire. Pour plus d'informations sur les options de configuration, consultez la section suivante.
`$listeners`	array	Non	Tableau d'objets `TransferListener` . Le SDK clone chacun d'eux `TransferListener` par fichier.
`$progressTracker`	`TransferListener`	Non	Un outil de suivi de la progression de tous les téléchargements.

`$config`options pour **`DownloadDirectoryRequest`**

Option	Type	Par défaut	Description
`s3_prefix`	chaîne	"	Préfixe pour le filtrage (si ce n'est pas le cas`list_objects_v2_args`).

Option	Type	Par défaut	Description
`track_progress`	bool	`false`	S'il faut suivre les progrès.
`filter`	appelable	`null`	Fonction permettant de filtrer les objets S3 à télécharger.
`download_object_request_modifier`	appelable	`null`	Fonction permettant de personnaliser chaque demande de téléchargement.
`list_objects_v2_args`	array	[]	Arguments de l'`ListObjectsV2` opération. Cette opération récupère les métadonnées des objets à télécharger.
`fails_when_destination_exists`	bool	false	S'il faut échouer lorsqu'un chemin de fichier de destination d'un objet existe déjà.
`failure_policy`	appelable	null	Fonction permettant de gérer les échecs de téléchargement.
`max_concurrency`	int	100	Le nombre maximal de téléchargements simultanés.
`target_part_size_bytes`	int	varie	Taille de pièce pour les téléchargements en plusieurs parties lorsque le type de téléchargement en plusieurs parties est varié.

Types d'options appelables **`$config`**

Nom d'option	nom du paramètre	type de paramètre	informations sur les paramètres
`filter`	`$objectKey`	chaîne	Le nom de la clé de l'objet indiqué dans le compartiment.
`download_object_request_modifier`	`$requestArgs`	array	Les arguments de demande pour chaque demande de téléchargement individuelle. Pour plus d'informations sur les options de tableau, consultez la section sur la syntaxe des paramètres de la méthode GetObject.
`failure_policy`	`$downloadObjectRequestArgs`	array	Les arguments de demande pour chaque demande de téléchargement individuelle. Pour plus d'informations sur les options de tableau, consultez la section sur la syntaxe des paramètres de la méthode GetObject.
	`$downloadDirectoryRequest`	array	Un tableau contenant le compartiment source et le répertoire cible pour l'opérati

Nom d'option	nom du paramètre	type de paramètre	informations sur les paramètres
			on du répertoire de téléchargement.
	`$reason`	`Throwable`	Le support d'excepti on qui contient des informations sur la cause de l'échec.
	`$download Directory Result`	DownloadDirectoryR esult *<add link>*	Objet contenant le nombre d'objets téléchargés avec succès et le nombre d'objets ayant échoué.

Comment fonctionnent les opérations d'annuaire

Cette section explique comment le gestionnaire de transfert S3 gère les chemins de fichiers et les clés d'objet lors des opérations de répertoire.

Gestion du chemin de téléchargement

Lorsque vous chargez un répertoire, le SDK crée des clés d'objet S3 à partir de chemins de fichiers :

1. Calcule le chemin de chaque fichier par rapport au répertoire source
2. Ajoute la `s3_prefix` valeur si elle est spécifiée (ajoute automatiquement une barre oblique finale)
3. Remplace le séparateur de répertoire du système d'exploitation par le `s3_delimiter` (par défaut`/`)

L'`recursive`option contrôle si les sous-répertoires sont inclus. Lorsque `false` (par défaut), seuls les fichiers de premier niveau sont chargés. Lorsque tous `true` les fichiers des sous-répertoires sont téléchargés, la structure des répertoires est préservée dans les clés d'objet.

Note

La clé dans S3 pour chaque fichier téléchargé sera le préfixe s3 fourni, nul par défaut, plus le chemin relatif du fichier à partir du répertoire source spécifié. De plus, nous remplaçons le séparateur de répertoire dans le nom de clé résolu par le délimiteur s3 fourni, qui sera par défaut /.

Exemple 1

```
$sourceDirectory = "/opt/my-upload-directory";
$s3Prefix = "important/";
$s3Delimiter = "/"; // Default value
```

Structure des annuaires

```
/opt/my-upload-directory/
-- my-file1.txt
-- my-file-2.txt
-- sub-dir/
---- my-another-file1.txt
---- my-another-file2.txt
```

Les clés de chaque fichier seront les suivantes :

```
$s3Prefix + $fileRelativePath;
```

```
- important/my-file1.txt
- important/my-file-2.txt
- important/sub-dir/my-another-file1.txt
- important/sub-dir/my-another-file2.txt
```

Exemple 2

```
$sourceDirectory = "/opt/my-docs";
$s3Prefix = ''; // No provided and it will defaulted to empty string
$s3Delimiter = "/"; // Default value
```

Structure des annuaires

```
/opt/my-docs/
-- my-file1.txt
-- my-file-2.txt
-- sub-dir/
---- my-another-file1.txt
---- my-another-file2.txt
```

Les clés de chaque fichier seront les suivantes :

```
$s3Prefix + $fileRelativePath;
```

```
- my-file1.txt
- my-file-2.txt
- sub-dir/my-another-file1.txt
- sub-dir/my-another-file2.txt
```

Example Gestion du chemin du répertoire de téléchargement

```
<?php
// Source directory: /home/user/photos/
//   fla/beach.jpg
//   fla/sunset.jpg
//   fla/pool/party.jpg
// With s3_prefix: '2023' and recursive: true
// Results:
//   2023/fla/beach.jpg
//   2023/fla/sunset.jpg
//   2023/fla/pool/party.jpg
```

Gestion du chemin de téléchargement

Lorsque vous téléchargez un répertoire, le SDK crée des chemins de fichiers locaux à partir des clés d'objet S3 :

1. Supprime la clé `s3_prefix` de l'objet si elle est spécifiée
2. Remplace le séparateur S3 (`/`) par le séparateur de répertoire du système d'exploitation

3. Ajoute le résultat au répertoire de destination

Les téléchargements sont toujours récursifs. Le SDK récupère tous les objets sous le préfixe spécifié et crée des sous-répertoires selon les besoins. Cela confirme que les chemins ne sont pas résolus en dehors du répertoire de destination pour des raisons de sécurité.

Example Gestion des chemins de répertoire de téléchargement

```
<?php
// S3 objects:
//   2023/fla/beach.jpg
//   2023/fla/sunset.jpg
//   2023/fla/pool/party.jpg
// With s3_prefix: '2023' and destination: /home/user/downloads
// Results:
//   /home/user/downloads/fla/beach.jpg
//   /home/user/downloads/fla/sunset.jpg
//   /home/user/downloads/fla/pool/party.jpg
```

Note

Le chemin du fichier pour les objets téléchargés sera résolu comme suit :

- Le préfixe s3, s'il est fourni, sera supprimé de la clé d'objet.
- Si le s3Delimiter est différent du séparateur de répertoire du système d'exploitation, le délimiteur s3 sera remplacé par le séparateur de répertoire du système d'exploitation.

Exemple 1

```
$s3Prefix = "my-prefix";
$s3Objects = [
    "my-prefix/file-1.txt",
    "my-prefix/file-2.txt",
    "my-prefix/subdir/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Une fois ces objets téléchargés, ils seront placés comme suit :

```
/opt/bucket/downloads/
```

```
-- file-1.txt
-- file-2.txt
-- subdir/
---- file-1.txt
```

Comme nous pouvons le constater, le préfixe a été supprimé du chemin de fichier final pour les objets.
Exemple 2 - sans préfixe s3

```
$s3Prefix = ""; // No provided
$s3Objects = [
    "README.md",
    "my-docs/file-1.txt",
    "my-docs/file-2.txt",
    "my-docs/statements/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Une fois ces objets téléchargés, ils seront placés comme suit :

```
/opt/bucket/downloads/
-- README.md
-- my-docs/
---- file-1.txt
---- file-2.txt
---- statements/
------ file-1.txt
```

Transférer des auditeurs

Vous pouvez implémenter votre propre écouteur qui réagit aux différents événements de transfert. Les événements sont les suivants :

- **`transferInitiated`**: Lorsqu'un transfert est lancé
- **`bytesTransferred`**: lorsqu'un transfert transfère des octets de la source vers la destination
- **`transferComplete`**: Lorsqu'un transfert se termine sans erreur
- **`transferFail`**: lorsqu'un transfert a échoué avant d'être terminé

Chaque `S3TransferManager` opération (par exemple, `uploadFile` et`downloadDirectory`) possède un `listeners` paramètre. Ce paramètre est un tableau qui s'attend à ce que chaque élément soit une instance de`AbstractTransferListener`.

`AbstractTransferListener`est une classe de base abstraite que vous pouvez étendre pour surveiller les opérations de transfert. Il fournit des implémentations de méthodes vides pour tous les événements de transfert. Il vous suffit de remplacer les méthodes pour les événements que vous souhaitez gérer.

Chaque `AbstractTransferListener` méthode reçoit un `$context` argument. Cet argument contient des informations relatives à la valeur clé de l'opération. Le contexte inclut :

- `request_args`(array) - Paramètres de demande
- `progress_snapshot`(instance de `TransferProgressSnapshot`*<add link>*) - État actuel de la demande
- `reason`(instance de`Throwable`) - fourni uniquement pour la `transferFail` méthode

La `progress_snapshot` propriété contient également la `reason` propriété en cas de défaillance. Vous pouvez accéder à l'exception via `$context['reason']` ou via`$context['progress_snapshot']#getReason()`.

Le code suivant montre la structure des `AbstractTransferListener` classes :

```
<?php

namespace Aws\S3\S3Transfer\Progress;

abstract class AbstractTransferListener
{
    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the request initialization.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferInitiated(array $context): void {}

    /**
```

```
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function bytesTransferred(array $context): bool {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferComplete(array $context): void {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *    as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     * - reason: (Throwable) The exception originated by the transfer failure.
     *
     * @return void
     */
    public function transferFail(array $context): void {}
}
```

L'exemple suivant montre une `AbstractTransferListener` implémentation qui remplace uniquement les `transferFail` méthodes `transferInitiated` et :

```
<?php

namespace MyApp\Listeners;

use Aws\S3\S3Transfer\Progress\AbstractTransferListener;

class MyCustomListener extends AbstractTransferListener
{
    public function transferInitiated(array $context): void
```

```
    {
        $progressSnapshot = $context['progress_snapshot'];

        echo "Transfer initiated for object with identifier: "
            . $progressSnapshot->getIdentifier();
    }

    public function transferFail(array $context): void
    {
        $progressSnapshot = $context['progress_snapshot'];
        $reason = $context['reason'];

        echo "Transfer for object with identifier: "
            . $progressSnapshot->getIdentifier() . " has failed.";

        echo "Completion Status: " . $progressSnapshot->getTransferredBytes()
            . "/"
            . $progressSnapshot->getTotalBytes() . " B";

        echo "Reason of failure is: " . $reason;
    }
}
```

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use MyApp\Listeners\MyCustomListener;

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


// The following example uses `MyCustomListener` in an `upload` operation.
$transferManager = new S3TransferManager(
    null,
    ['default_region' => 'us-east-2']
);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        source: "/tmp/myfile.txt",
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'MyKey'
```

```
        ],
        listeners: [
            new MyCustomListener()
        ]
    )
);

$result = $uploadPromise->wait();
```

Suivi des progrès

S3 Transfer Manager fournit des fonctionnalités intégrées de suivi de la progression.

Suivi de toutes les opérations

Vous pouvez activer le suivi pour toutes les opérations (fichier et répertoire) en configurant `track_progress` cette option `true` sur l'`S3TransferManager`instance.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(
    null,
    [
        'track_progress' => true,
    ]
);
```

Avec ce paramètre, le SDK utilise le paramètre intégré `SingleProgressTracker` *<add link>* pour les opérations sur les fichiers et utilise le `MultiProgressTracker` <add link>pour les opérations de répertoire.

Suivi d'une opération sur un seul fichier

Si vous n'avez pas activé le suivi de la progression dans le gestionnaire de transferts, vous pouvez activer le suivi selon une méthode individuelle de deux manières.

- `track_progress`Défini sur `true` dans le `$config` tableau pour une opération de fichier spécifique.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [   // $config array
            'track_progress' => true,
        ],
        []
    )
);
```

Créez une nouvelle `SingleProgressTracker` instance ou fournissez une implémentation personnalisée de`AbstractTransferListener`. Transmettez votre tracker aux opérations sur les fichiers en tant que `$progressTracker` paramètre.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/Myfile.txt',  // Or an instance of StreamInterface.
```

```
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'file.txt',
        ],
        [],
        [],
        new SingleProgressTracker()  // Or custom implementation of the
 TransferLister interface.
    )
);
```

L'exemple suivant montre un exemple de sortie de console depuis le module intégré `SingleProgressTracker` pour la progression du téléchargement d'un fichier nommé `MyFile.txt` :

```
MyFile.txt:
[#########################                                  ] 50% 5242880/10485760 B
```

Suivi d'une seule opération d'annuaire

Si vous n'avez pas activé le suivi de la progression dans le gestionnaire de transferts, vous pouvez l'activer pour les opérations d'annuaire de deux manières.

Définissez « track_progress » sur `true` dans l'argument du `$config` tableau de la méthode.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
```

```
        [   // $config array
            'track_progress' => true,
        ],
        []
    )
);
```

Créez une nouvelle `MultiProgressTracker` instance et transmettez-la aux opérations d'annuaire en tant que `$progressTracker` paramètre.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\Progress\MultiProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [],
        [],
        new MultiProgressTracker()
    )
);
```

L'exemple suivant montre un exemple de sortie de console depuis le module intégré `MultiProgressTracker` pour la progression du téléchargement d'un répertoire :

```
MyFile.4.txt:
[##################################################] 100% 4194304/4194304 B
MyFile.5.txt:
[##################################################] 100% 24117248/24117248 B
MyFile.2.txt:
[                                                  ] 0% 0/10485760 B
MyFile.3.txt:
[##################################################] 100% 18874368/18874368 B
MyFile.1.txt:
[##################################################] 100% 1048576/1048576 B
--------------------------------------------------
[##########################################         ] 80% Completed: 3/5, Failed: 0/5
```

La progression de chaque fichier est répertoriée ainsi que le résumé de la progression dans la dernière ligne.

Suivi personnalisé de la progression

Vous pouvez implémenter votre propre suivi de progression pour les opérations sur les fichiers. Créez une classe qui étend l'`AbstractTransferListener`interface et transmettez-la à la méthode sur la demande.

L'implémentation suivante affiche les informations de progression au format texte pour une seule opération sur un fichier :

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\AbstractTransferListener;
use Aws\S3\S3Transfer\S3TransferManager;

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

class MyProgressTracker extends AbstractTransferListener
{
    public function transferInitiated(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Starting transfer of {$key}...\n";
    }

    public function bytesTransferred(array $context): bool
    {
        $snapshot = $context['progress_snapshot'];
```

```
        $percent = round($snapshot->ratioTransferred() * 100, 2);
        $transferred = round($snapshot->getTransferredBytes() / 1024 / 1024, 2);
        $total = round($snapshot->getTotalBytes() / 1024 / 1024, 2);

        echo "Progress: {$percent}% ({$transferred}MB of {$total}MB)\n";
    }

    public function transferComplete(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Transfer complete for {$key}!\n";
    }

    public function transferFail(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        $reason = $context['reason']->getMessage();
        echo "Transfer failed for {$key}: {$reason}\n";
    }
}

$transferManager = new S3TransferManager(
    null, []
);

// Use your custom tracker.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt', // Or an instance of StreamInterface.
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ],
        [],
        [],
        new MyProgressTracker()
    )
);
```

L'exemple suivant montre un exemple de sortie vers la console pour le téléchargement de fichiers en cours :

```
Starting transfer of file.txt...
```

```
Progress: 42% (16.8MB of 40MB)
```

Personnalisez un outil de suivi de progression intégré

Vous pouvez personnaliser le suivi de la progression de différentes manières :

- Créez une implémentation personnalisée en étendant `AbstractTransferListener` et en implémentant `ProgressTrackerInterface`*<add link>*.
- Personnalisez les implémentations existantes en configurant celles `ConsoleProgressBar` *<add link>* qui sont transmises aux trackers intégrés.
- Implémentez une personnalisation `ProgressBar` *<add link>* et transmettez-la aux trackers intégrés.

Les exemples suivants montrent comment personnaliser le suivi de la progression. Reportez-vous au diagramme de classes pour voir la relation entre les composants de suivi de la progression dans le SDK.

Personnalisez un **`ConsoleProgressBar`**

Personnaliser le format de barre par défaut

L'exemple suivant personnalise le format de barre de progression par défaut, qui est `ColoredTransferProgressBarFormat`*<add link>*. Ce format affiche la progression avec différentes couleurs en fonction de l'état du transfert.

L'exemple change :

- le caractère compris entre « # » et « = » par défaut
- largeur de barre comprise entre 50 et 100

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

```
$progressBar = new ConsoleProgressBar(
    progressBarChar: '=',
    progressBarWidth: 100,
// Default `ColoredTransferProgressBarFormat` is used.
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file.txt";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

L'image suivante montre la sortie de console pour un module intégré personnalisé `ConsoleProgressBar` d'une largeur de 100 et d'un caractère « = » :



Personnalisez en utilisant le format de barre de progression simple

L'exemple suivant personnalise la barre de progression par défaut en utilisant la barre intégrée `PlainProgressBarFormat` *<add link>* avec un caractère et une largeur personnalisés. Les `PlainProgressBarFormat` écrans sont uniquement en noir. La couleur ne change pas en fonction de l'état du transfert.

L'exemple change :

- le caractère compris entre « # » et « * » par défaut
- largeur de barre comprise entre 50 et 25
- l'`progressBarFormat`instance par défaut comprise entre `ColoredTransferProgressBarFormat` et `PlainProgressBarFormat`

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\PlainProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3TransferManager;

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

$progressBar = new ConsoleProgressBar(
    progressBarChar: '*',
    progressBarWidth: 25,
    progressBarFormat: new PlainProgressBarFormat(),
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file.txt";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

L'image suivante montre la sortie de console pour une version intégrée personnalisée `ConsoleProgressBar` utilisant un format de barre de progression simple d'une largeur de 25 et d'un caractère « * » :



Personnalisez en créant un nouveau format de barre

L'exemple suivant personnalise la barre de progression par défaut en utilisant un nouveau format de barre (extension `ProgressBarFormat`*<add link>*) au lieu de personnaliser un format intégré.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\ProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3TransferManager;

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

// Implement a custom bar format.
class PercentBarFormat extends ProgressBarFormat
{
    public const FORMAT_TEMPLATE = "|object_name| - |percent|% [|transferred| |unit|
ytes]";
    public const FORMAT_PARAMETERS = [
        'object_name',       // `uploadRequestArgs` Key parameter.
        'percent',           //  Percent transferred.
        // 'progress_bar',      Optional, if used, default is a `ConsoleProgressBar`.
        // 'to_be_transferred', Optional.
        'transferred',       //  Default is bytes transferred.
        'unit',              //  Default is 'B'.
```

```
    ];

    public function getFormatTemplate(): string
    {
        return self::FORMAT_TEMPLATE;
    }

    public function getFormatParameters(): array
    {
        return self::FORMAT_PARAMETERS;
    }

    protected function getFormatDefaultParameterValues(): array
    {
        return [];
    }
}

// Create an instance of the custom implemention
$progressBarFormat = new PercentBarFormat();
$progressBar = new ConsoleProgressBar(
    progressBarFormat: $progressBarFormat,
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

L'image suivante montre la sortie de console pour une version intégrée personnalisée `ConsoleProgressBar` utilisant un format de barre de progression personnalisé :

```
key – 0% [0 Bytes]
key – 11% [1153434 Bytes]
key – 55% [5767168 Bytes]
key – 100% [10485760 Bytes]
```

Un format de barre personnalisé accepte plusieurs paramètres :

- nom d'objet
- pourcentage
- barre de progression
- à transférer
- transférées
- unité

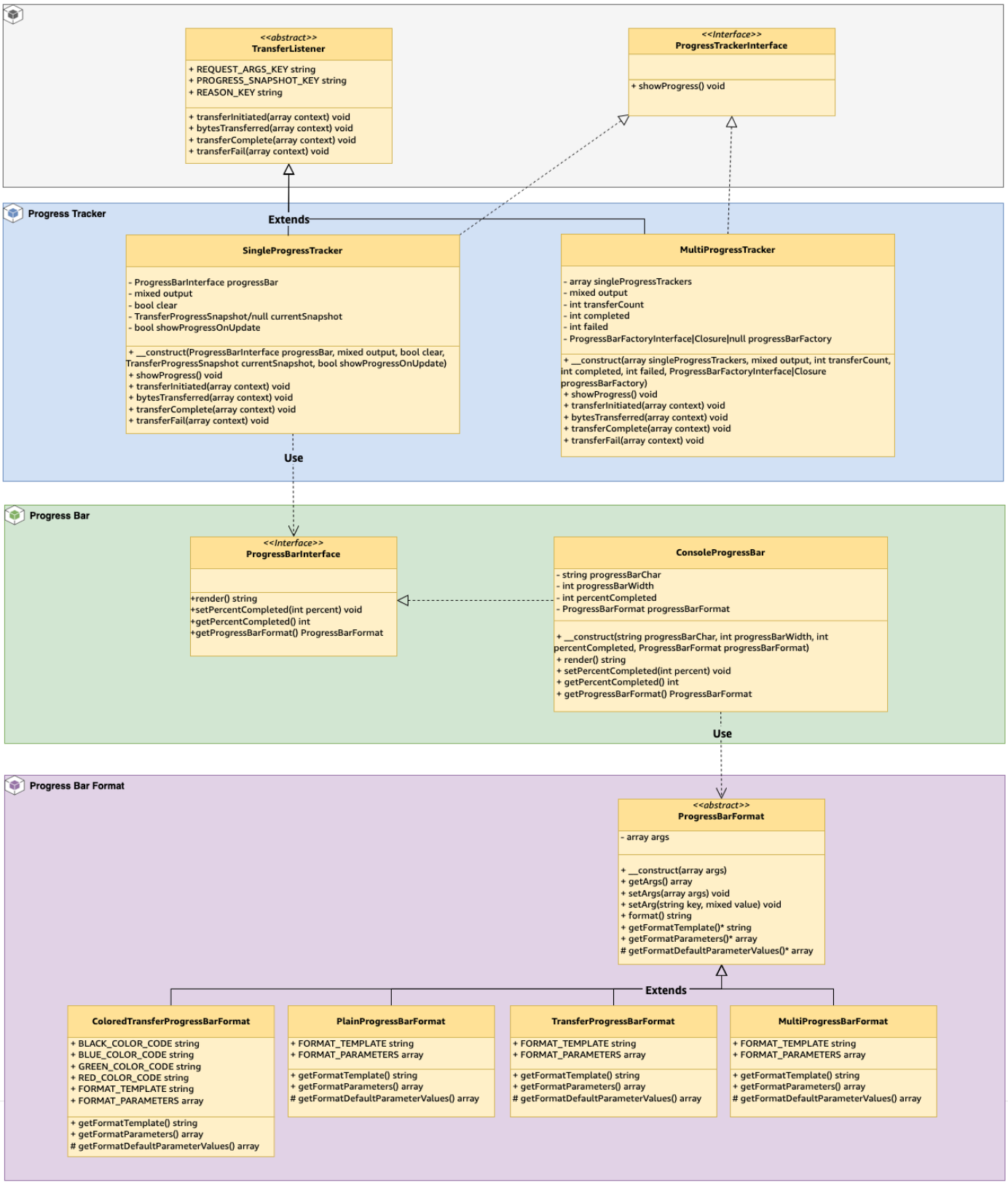
Vous pouvez combiner les noms de paramètres délimités (caractères en forme de tube) pour personnaliser la sortie. L'exemple utilise le modèle suivant. La `getFormatTemplate` méthode de la `PercentBarFormat` classe renvoie ce modèle :

```
"|object_name| - |percent|% [|transferred| |unit|ytes]"
```

Diagramme de classes : composants de suivi de S3 Transfer Manager

Le diagramme de classes suivant montre comment les composants de suivi fonctionnent ensemble dans le gestionnaire de transfert S3. La compréhension de ces relations vous permet de mettre en œuvre des écouteurs personnalisés et de suivre les progrès.

S3 Transfer Manager tracking components



Gestion des erreurs

S3 Transfer Manager utilise des promesses pour les opérations asynchrones. Vous pouvez gérer les erreurs à l'aide de la `then` méthode `otherwise` ou de la promesse. Vous pouvez également encapsuler votre implémentation dans un `try-catch` bloc.

Gestion des erreurs de promesse

En utilisant `otherwise` :

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);

$uploadPromise->otherwise(function (Throwable $reason) {
    echo "Upload failed: " . $reason->getMessage() . "\n";
});
```

En utilisant `then` :

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Models\UploadResult;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
```

```
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);


$uploadPromise->then(
    function (UploadResult $result) {
        echo "Upload succeeded!\n";
    },
    function (Throwable $error) {
        echo "Upload failed: " . $error->getMessage() . "\n";
    }
)->wait();
```

Utilisation du bloc try-catch

```
<?php

use Aws\S3\S3Transfer\Exception\S3TransferException;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);

try {
```

```
    $uploadPromise->wait();
} catch (S3TransferException $exception) {
    echo "Upload failed: " . $exception->getMessage() . "\n";
}
```

<Add more details about S3TransferManager exception and also a link to the API reference for it.>

Politiques relatives aux défaillances des opérations d'annuaire

Pour les opérations d'annuaire, vous pouvez également spécifier un rappel de politique d'échec. Ce rappel gère les échecs de fichiers individuels et décide si le téléchargement du répertoire doit se poursuivre.

Example de l'utilisation d'une fonction « failure_policy » pour le fonctionnement **`uploadDirectory`**

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $uploadDirectoryRequestArgs,
                $reason,
                $uploadDirectoryResponse
            ) {
                echo "Failed to upload {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}\n";
                echo "So far, uploaded: " .
                    "{$uploadDirectoryResponse->getObjectsUploaded()}, " .
                    "failed: {$uploadDirectoryResponse->getObjectsFailed()}\n";
```

```
                // Return true to continue with other files,
                // or throw an exception to abort.
                return true;
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Example de l'utilisation d'une fonction « failure_policy » pour le fonctionnement **downloadDirectory**

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Log errors but continue with other files.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/directory',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $downloadDirectoryRequestArgs,
                $reason,
                $downloadDirectoryResponse
            ) {
                error_log("Failed to download {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}");

                // If we've had too many failures, abort the entire operation.
                if ($downloadDirectoryResponse->getObjectsFailed() > 10) {
                    throw new \Exception(
                        "Too many download failures, aborting operation"
                    );
                }
```

```
                // Return void to continue with other files.
                return;
            },
        ]
    )
);
$downloadDirPromise->wait();
```

Utilisation avancée

Cette section couvre les modèles et techniques d'utilisation avancés du gestionnaire de transfert S3.

Chargements partitionnés

S3 Transfer Manager utilise automatiquement des téléchargements partitionnés pour les fichiers volumineux. Vous pouvez personnaliser ce comportement à l'aide des options de configuration.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/large/file.mp4',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'videos/large-file.mp4',
        ],
        [
            // Use multipart upload for files larger than 100MB.
            'multipart_upload_threshold_bytes' => 100 * 1024 * 1024,

            // Use 25MB parts for multipart uploads.
            'target_part_size_bytes'           => 25 * 1024 * 1024,
        ]
    )
);
```

```
$uploadPromise->wait();
```

Téléchargements en plusieurs parties

Vous pouvez personnaliser le comportement de téléchargement en plusieurs parties.

```
<?php

use Aws\S3\S3Transfer\AbstractMultipartDownloader;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$downloadPromise = $transferManager->download(
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/large-file.mp4',
        [],
        [
            // Use 25MB parts for multipart downloads.
            'target_part_size_bytes'  => 25 * 1024 * 1024,

            // Use ranged-based download instead of part-based.
            'multipart_download_type' =>
 AbstractMultipartDownloader::RANGED_GET_MULTIPART_DOWNLOADER,
        ]
    )
);
$downloadPromise->wait();
```

Pour plus d'informations sur les autres membres de la `AbstractMultipartDownloader` classe, consultez la documentation de l'API*<add link>*.

Filtres personnalisés

Vous pouvez utiliser les fonctions de filtrage pour charger ou télécharger des fichiers de manière sélective pour les opérations de répertoire.

Filtre de répertoire de téléchargement

Pour plus d'informations sur les informations sur les paramètres, consultez l'option de filtre appelable de la `uploadDirectory` méthode.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Upload files modified in the last 24 hours.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'filter' => function ($file) {
                $modTime = filemtime($file);
                return (time() - $modTime) < 86400; // 24 hours
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Filtre de répertoire de téléchargement

Pour plus d'informations sur les informations sur les paramètres, consultez l'option de filtre appelable de la `downloadDirectory` méthode.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
```

```
// Download files with a specific prefix.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/directory',
        [],
        [
            'filter' => function ($key) {
                return strpos($key, 'reports/2023/') === 0;
            },
        ]
    )
);
$downloadDirPromise->wait();
```

Rappels préalables à la demande

Vous pouvez modifier les paramètres de demande pour chaque fichier à l'aide de rappels préalables à la demande. Pour plus d'informations sur les paramètres, consultez les rubriques suivantes :

- option appelable put_object_request_callaback de la méthode `uploadDirectory`
- option appelable download_object_request_modifier de la méthode `downloadDirectory`

Example d'un rappel de pré-demande pour la méthode **`uploadDirectory`**

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Set custom metadata for each uploaded file.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
```

```
            'upload_object_request_modifier' => function ($args) {
                $extension = pathinfo($args['Key'], PATHINFO_EXTENSION);

                switch ($extension) {
                    case 'jpg':
                    case 'jpeg':
                        $args['ContentType'] = 'image/jpeg';
                        break;
                    case 'png':
                        $args['ContentType'] = 'image/png';
                        break;
                    case 'pdf':
                        $args['ContentType'] = 'application/pdf';
                        break;
                }

                $args['Metadata'] = [
                    'uploaded_at' => date('Y-m-d H:i:s'),
                ];
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Ready-to-use exemples

Cette section fournit une classe d'assistance complète avec des méthodes statiques pour les opérations courantes de S3 Transfer Manager. La classe d'assistance simplifie l'utilisation. Il gère l'initialisation et la configuration en interne.

TransferHelper Classe S3

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use Psr\Http\Message\StreamInterface;
```

```
class S3TransferHelper
{
    /**
     * Upload a file from a local path to S3.
     */
    public static function uploadFile(
        string $filePath,
        string $bucket,
        string $key
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $filePath,
            [
                'Bucket' => $bucket,
                'Key'    => $key,
            ],
            []
        );

        return $transferManager->upload($uploadRequest)->wait()->getResult();
    }

    /**
     * Upload from a stream to S3
     */
    public static function uploadFromStream(
        StreamInterface $stream,
        string          $bucket,
        string          $key,
        bool            $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $stream,
            [
                'Bucket' => $bucket,
                'Key'    => $key,
            ],
            ['track_progress' => $trackProgress]
```

```
        );

        return $transferManager->upload($uploadRequest)->wait()->getResult();
    }

    /**
     * Download a file from S3 to a local path
     */
    public static function downloadFile(
        string $bucket,
        string $key,
        string $destinationPath,
        bool   $failsWhenDestinationExists = false,
        bool   $trackProgress = true,
    ): void
    {
        $transferManager = new S3TransferManager();
        $downloadRequest = new DownloadRequest(
            [
                'Bucket' => $bucket,
                'Key'    => $key
            ],
            [],
            ['track_progress' => $trackProgress]
        );

        $downloadFileRequest = new DownloadFileRequest(
            $destinationPath,
            $failsWhenDestinationExists,
            $downloadRequest
        );

        $transferManager->downloadFile($downloadFileRequest)->wait();
    }

    /**
     * Upload an entire directory to S3
     */
    public static function uploadDirectory(
        string $sourceDirectory,
        string $bucket,
        string $s3Prefix = '',
        bool   $trackProgress = true,
    ): array
```

```
    {
        $transferManager = new S3TransferManager();
        $uploadDirectoryRequest = new UploadDirectoryRequest(
            $sourceDirectory,
            $bucket,
            [],
            [
                's3_prefix'      => $s3Prefix,
                'track_progress' => $trackProgress,
            ]
        );

        $result = $transferManager->uploadDirectory($uploadDirectoryRequest)->wait();

        return [
            'uploaded' => $result->getObjectsUploaded(),
            'failed'   => $result->getObjectsFailed(),
        ];
    }

    /**
     * Download directory from S3
     */
    public static function downloadDirectory(
        string $bucket,
        string $destinationDirectory,
        string $s3Prefix = '',
        bool   $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();
        $downloadDirectoryRequest = new DownloadDirectoryRequest(
            $bucket,
            $destinationDirectory,
            [],
            [
                's3_prefix'      => $s3Prefix,
                'track_progress' => $trackProgress,
            ]
        );

        $result = $transferManager->downloadDirectory($downloadDirectoryRequest)-
>wait();
```

```
        return [
            'downloaded' => $result->getObjectsDownloaded(),
            'failed'     => $result->getObjectsFailed(),
        ];
    }
}
```

Exemples d'utilisation de la classe d'assistance

Les exemples suivants montrent comment utiliser la `S3TransferHelper` classe.

```
<?php

require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/S3TransferHelper.php';

// Upload a local file
$result = S3TransferHelper::uploadFile(
    '/path/to/local/document.pdf',
    'amzn-s3-demo-bucket',
    'documents/document.pdf'
);

echo "File uploaded successfully. ETag: " . $result['ETag'] . "\n";

// Download a file to local path
S3TransferHelper::downloadFile(
    'amzn-s3-demo-bucket',
    'documents/document.pdf',
    '/path/to/local/downloaded-document.pdf',
    false, // Don't fail if destination exists
    true   // Track progress
);

echo "File downloaded successfully!\n";

// Upload entire directory
$result = S3TransferHelper::uploadDirectory(
    '/path/to/local/photos',
    'amzn-s3-demo-bucket',
    'photos/vacation-2023/', // S3 prefix
    true // Track progress
);
```

```
echo "Directory upload completed.\n";
echo "Uploaded: {$result['uploaded']} files\n";
echo "Failed: {$result['failed']} files\n";

// Download directory from S3
$result = S3TransferHelper::downloadDirectory(
    'amzn-s3-demo-bucket',
    '/path/to/local/downloads',
    'photos/vacation-2023/', // S3 prefix to download
    true // Track progress
);

echo "Directory download completed.\n";
echo "Downloaded: {$result['downloaded']} files\n";
echo "Failed: {$result['failed']} files\n";
```

Transférez des annuaires vers et depuis Amazon S3 à l'aide de la AWS SDK pour PHP version 3

Dans la AWS SDK pour PHP version 3, vous utilisez la `Transfer` classe pour charger des répertoires entiers dans un compartiment Amazon S3 et télécharger des compartiments entiers dans un répertoire local.

Téléchargement d'un répertoire local sur Amazon S3

L'`Aws\S3\Transfer`objet effectue les transferts. L'exemple suivant montre comment charger de manière récursive un répertoire local de fichiers dans un compartiment Amazon S3.

```
// 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.
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);

// Perform the transfer synchronously.
```

```
$directoryTransfer->transfer();
```

Dans cet exemple, nous avons créé un client Amazon S3, créé un `Transfer` objet et effectué le transfert de manière synchrone.

L'exemple précédent montre la quantité minimale de code nécessaire pour effectuer un transfert. Un `Transfer` objet peut également effectuer des transferts de manière asynchrone et possède différentes options de configuration que vous pouvez utiliser pour personnaliser les transferts.

Vous pouvez télécharger les fichiers locaux dans un « sous-dossier » d'un compartiment Amazon S3 en fournissant un préfixe de clé dans `s3://` l'URI. L'exemple suivant télécharge les fichiers locaux sur le disque pour le compartiment `bucket` et stocke les fichiers sous le préfixe de clé `foo`.

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

Téléchargement d'un compartiment Amazon S3

Vous pouvez télécharger de manière récursive un compartiment Amazon S3 vers un répertoire local sur disque en spécifiant l'`$source`argument sous forme d'URI Amazon S3 (par exemple,`s3://bucket`) et l'`$dest`argument comme chemin d'accès à un répertoire local.

```
// Where the files will be sourced from.
$source = 's3://bucket';

// Where the files will be transferred to.
$dest = '/path/to/destination/dir';

$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

> **Note**
>
> Le SDK crée automatiquement tous les répertoires nécessaires lors du téléchargement des objets du compartiment.

Vous pouvez inclure un préfixe de clé dans l'URI Amazon S3 après le compartiment pour télécharger uniquement les objets stockés dans un « pseudo-dossier ». L'exemple suivant télécharge uniquement les fichiers stockés sous le préfixe de clé « /foo » du compartiment donné.

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

Configuration

Le constructeur de l'objet `Transfer` accepte les arguments suivants.

`$client`

L'objet `Aws\ClientInterface` à utiliser pour effectuer les transferts.

`$source`(chaîne |**`Iterator`**)

Les données source concernées par le transfert. Cela peut pointer vers un chemin local sur le disque (par exemple`/path/to/files`) ou vers un compartiment Amazon S3 (par exemple,`s3://bucket`). L'URI `s3://` peut également contenir un préfixe de clé qui peut être utilisé pour transférer uniquement les objets dotés d'un préfixe commun.

Si l'`$source`argument est un URI Amazon S3, il doit être un répertoire local (et vice versa). `$dest`

En plus de fournir une valeur de chaîne, vous pouvez également fournir un objet `\Iterator` qui génère des noms de fichier absolus. Si vous fournissez un `\Iterator` objet, vous devez fournir une `base_dir` option dans le tableau `$options` associatif.

`$dest`

La destination de transfert des fichiers. Si l'`$source`argument est un chemin local sur le disque, `$dest` il doit s'agir d'un URI de compartiment Amazon S3 (par exemple,`s3://bucket`). Si l'`$source`argument est un URI de compartiment Amazon S3, il doit être un chemin local sur le disque. `$dest`

`$options`

Un tableau associatif d’options de transfert. Les options de transfert suivantes sont valides :

`add_content_md5` (bool)

Réglez sur `true` pour calculer le MD5 checksum pour les téléchargements.

`base_dir` (chaîne)

Répertoire de base de la source, si `$source` est un itérateur. Si l'option `$source` n'est pas un tableau, cette option est ignorée.

`before` (joignable)

Un rappel à invoquer avant chaque transfert. Les rappels doivent avoir une signature de fonction telle que `function (Aws\Command $command) {...}`. La commande fournie est une commande `GetObject`, `PutObject`, `CreateMultipartUpload`, `UploadPart` ou `CompleteMultipartUpload`.

`mup_threshold` (int)

Taille en octets pour laquelle il est préférable d'utiliser un chargement partitionné au lieu de `PutObject`. Sa valeur par défaut est de `16777216` (16 Mo).

`concurrency` (int, default=5)

Nombre de fichiers à charger simultanément. La valeur de simultanéité idéale varie en fonction du nombre de fichiers en cours de chargement et de la taille moyenne de chaque fichier. En général, les petits fichiers bénéficient d'une simultanéité plus élevée, contrairement aux fichiers plus volumineux.

`debug` (bool)

Définissez ce paramètre sur `true` pour imprimer les informations de débogage pour les transferts. Définissez ce paramètre sur une ressource `fopen()` pour écrire sur un flux spécifique au lieu d'écrire sur STDOUT.

Transferts asynchrones

L'objet `Transfer` est une instance de `GuzzleHttp\Promise\PromisorInterface`. Cela signifie que le transfert peut se produire de façon asynchrone et est initié en appelant la méthode de l'objet `promise`.

```
$source = '/path/to/source/files';
$dest = 's3://bucket';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
```

```
// Initiate the transfer and get a promise.
$promise = $directoryTransfer->promise();

// Do something when the transfer is complete using the then() method.
$promise->then(function () {
    echo 'Done!';
});
```

La promesse est rejetée si le transfert d'un fichier échoue. Vous pouvez gérer les transferts en échec de façon asynchrone à l'aide de la méthode `otherwise` de la promesse. La fonction `otherwise` accepte un rappel pour invoquer lorsqu'une erreur se produit. Le rappel accepte le motif `$reason` de rejet, qui est généralement une instance de `Aws\Exception\AwsException` (bien qu'une valeur de n'importe quel type puisse être délivrée au rappel).

```
$promise->otherwise(function ($reason) {
    echo 'Transfer failed: ';
    var_dump($reason);
});
```

Etant donné que l'objet `Transfer` renvoie une promesse, ces transferts peuvent se produire simultanément avec d'autres promesses asynchrones.

Personnalisation du transfert d'annuaire

En ajoutant un rappel au constructeur, vous pouvez personnaliser les options qu'il exécute. `Transfer`

```
$uploader = new Transfer($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';
        }
    },
```

```
]);
```

Chiffrement côté client Amazon S3 dans la AWS SDK pour PHP version 3

Avec le chiffrement côté client, les données sont chiffrées et déchiffrées directement dans votre environnement. Cela signifie que ces données sont chiffrées avant d'être transférées vers Amazon S3 et que vous ne comptez pas sur un service externe pour gérer le chiffrement à votre place. Pour les nouvelles implémentations, nous suggérons d'utiliser `S3EncryptionClientV3` et de `S3EncryptionMultipartUploaderV3` remplacer le `S3EncryptionClientV2` `S3EncryptionMultipartUploaderV2` et le obsolète `S3EncryptionClient` et. `S3EncryptionMultipartUploader` Il est recommandé que les anciennes implémentations utilisant toujours les versions obsolètes tentent de migrer. `S3EncryptionClientV3`maintient le support pour le déchiffrement des données chiffrées à l'aide de l'ancienne version. `S3EncryptionClient`

AWS SDK pour PHP implémente le chiffrement des enveloppes et utilise OpenSSL pour le chiffrement et le déchiffrement. L'implémentation est interopérable avec d'autres SDKs qui correspondent à ses fonctionnalités prises en charge. Elle est également compatible avec le kit SDK du flux de travail asynchrone basé sur les promesses.

Guide de migration

Pour ceux qui essaient de migrer des clients obsolètes vers les nouveaux clients, il existe un guide de migration pour migrer de la v1 vers la v2 ici et un guide de migration pour passer de la v2 à la v3 ici.

Configuration

Pour démarrer avec le chiffrement côté client, vous avez besoin des éléments suivants :

- Une cléAWS KMS de chiffrement
- Compartiment S3

Avant d'exécuter un exemple de code, configurez vos AWS informations d'identification. Voir Informations d'identification pour la AWS SDK pour PHP version 3.

Chiffrement

Le chargement d'un objet chiffré `S3EncryptionClientV3` nécessite quatre paramètres supplémentaires en plus des `PutObject` paramètres standard :

- `'@KmsEncryptionContext'`est une paire clé-valeur qui peut être utilisée pour ajouter une couche de sécurité supplémentaire à votre objet chiffré. Le client de chiffrement doit transmettre la même clé, ce qu'il fera automatiquement lors d'un appel get. Si aucun contexte supplémentaire n'est souhaité, transmettez un tableau vide.
- `@CipherOptions`sont des configurations supplémentaires pour le chiffrement, notamment le chiffrement à utiliser et la taille de la clé.
- `@MaterialsProvider`est un fournisseur qui gère la génération d'une clé de chiffrement et d'un vecteur d'initialisation, ainsi que le chiffrement de votre clé de chiffrement.
- `@CommitmentPolicy`est une option politique qui dicte la manière dont un objet est lu, soit avec un engagement clé, soit sans engagement clé, et comment un objet est écrit avec un engagement clé ou sans engagement clé.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV3;

 // Let's construct our S3EncryptionClient using an S3Client
 $encryptionClient = new S3EncryptionClientV3(
     new S3Client([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ])
 );

 $kmsKeyId = 'kms-key-id';
 $materialsProvider = new KmsMaterialsProviderV3(
     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,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> Outre Amazon S3 et les erreurs de service AWS KMS basées sur Amazon, vous pouvez recevoir des `InvalidArgumentException` objets renvoyés si vous n'`'@CipherOptions'` êtes pas correctement configuré.

Déchiffrement

Le téléchargement et le déchiffrement d'un objet comportent cinq paramètres supplémentaires, dont deux sont obligatoires, en plus des paramètres standard. `GetObject` Le client détectera les options de chiffrement de base pour vous.

- **`'@SecurityProfile'`**: si le paramètre est défini sur « V3 », seuls les objets chiffrés sont compatibles avec la version V3

  le format peut être déchiffré. La définition de ce paramètre sur « V3_AND_LEGACY » permet également de déchiffrer les objets chiffrés dans un format compatible avec la version 1. Pour prendre en charge la migration, définissez @ sur « SecurityProfile V3_AND_LEGACY ». Utilisez « V3 » uniquement pour le développement de nouvelles applications.
- **`'@MaterialsProvider'`**est un fournisseur qui gère la génération d'une clé de chiffrement et d'un vecteur d'initialisation, comme

  ainsi que le chiffrement de votre clé de chiffrement.

- **`'@KmsAllowDecryptWithAnyCmk'`**: (facultatif) La définition de ce paramètre sur true active le déchiffrement

  sans fournir d'identifiant de clé KMS au constructeur du MaterialsProvider. La valeur par défaut est false.

- **`'@CipherOptions'`**(facultatif) sont des configurations supplémentaires pour le chiffrement, notamment lesquelles

  chiffrement à utiliser et taille de clé.

- **`@CommitmentPolicy`**option de politique qui dicte la façon dont un objet est lu soit avec

  engagement clé ou sans engagement clé et comment un objet est écrit avec ou sans engagement clé.

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> Outre Amazon S3 et les erreurs de service AWS KMS basées sur Amazon, vous pouvez recevoir des `InvalidArgumentException` objets renvoyés si vous n'`'@CipherOptions'`êtes pas correctement configuré.

Configuration du chiffrement

`'Cipher'` (chaîne)

Méthode de chiffrement que le client de chiffrement utilise lors du chiffrement. Seul « gcm » est pris en charge pour le moment.

> **Important**
>
> PHP est mis à jour dans la version 7.1 pour inclure les paramètres supplémentaires nécessaires pour chiffrer et déchiffrer à l'aide d'OpenSSL pour le chiffrement GCM. Pour les versions 7.0 et antérieures de PHP, un polyfill pour le support GCM est fourni et utilisé par les clients `S3EncryptionClientV2` de chiffrement et. `S3EncryptionMultipartUploaderV2` Cependant, les performances pour les entrées volumineuses seront beaucoup plus lentes en utilisant le polyfill qu'en utilisant l'implémentation native de PHP 7.1+. Il peut donc être nécessaire de mettre à niveau les environnements des anciennes versions de PHP pour les utiliser efficacement.

`'KeySize'` (int)

Longueur de la clé de chiffrement de contenu à générer pour le chiffrement. Valeur par défaut : 256 bits. Les options de configuration valides sont 256 bits.

`'Aad'` (chaîne)

« Données d'authentification supplémentaires » facultatives à inclure à votre charge utile chiffrée. Ces informations sont validées sur déchiffrement. `Aad` est disponible uniquement lorsque vous utilisez le chiffrement « gcm ».

> **Important**
>
> Les données d'authentification supplémentaires ne sont pas prises en charge par tous AWS SDKs et il est donc possible que d'autres ne SDKs soient pas en mesure de déchiffrer les fichiers chiffrés à l'aide de ce paramètre.

Stratégies de métadonnées

Vous avez également la possibilité de fournir une instance d'une classe qui implémente `Aws\Crypto\MetadataStrategyInterface`. Cette interface simple gère l'enregistrement et le chargement de `Aws\Crypto\MetadataEnvelope` qui contient vos supports de chiffrement d'enveloppe. Le kit SDK fournit deux classes qui implémentent ceci : `Aws\S3\Crypto\HeadersMetadataStrategy` et `Aws\S3\Crypto\InstructionFileMetadataStrategy`. `HeadersMetadataStrategy` est utilisé par défaut.

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => $strategy,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@KmsEncryptionContext' => [],
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V3',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Les constantes de nom de classe `HeadersMetadataStrategy` et `InstructionFileMetadataStrategy` peuvent également être fournies en appelant ::class.

```
$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => HeadersMetadataStrategy::class,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> En cas d'échec après chargement d'un fichier d'instructions, il ne sera pas automatiquement supprimé.

Chargements partitionnés

Il est également possible d'exécuter un chargement partitionné avec un chiffrement côté client. `Aws\S3\Crypto\S3EncryptionMultipartUploaderV3`prépare le flux source pour le chiffrement avant le téléchargement. Sa création s'appuie sur une expérience similaire à l'utilisation de `Aws\S3\MultipartUploader` et de `Aws\S3\Crypto\S3EncryptionClientV3`. `S3EncryptionMultipartUploaderV3` peut traiter la même option `'@MetadataStrategy'` que `S3EncryptionClientV3`, ainsi que toutes les configurations `'@CipherOptions'` disponibles.

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    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 S3EncryptionMultipartUploaderV3(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
    [
        '@MaterialsProvider' => $materialsProvider,
```

```
        '@CipherOptions' => $cipherOptions,
        '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
        'bucket' => $bucket,
        'key' => $key,
    ]
);
$multipartUploader->upload();
```

> **Note**
>
> Outre Amazon S3 et les erreurs de service AWS KMS basées sur Amazon, vous pouvez recevoir des `InvalidArgumentException` objets renvoyés si vous n'`'@CipherOptions'` êtes pas correctement configuré.

Protection de l'intégrité des données avec des checksums

Amazon Simple Storage Service (Amazon S3) permet de spécifier une somme de contrôle lorsque vous chargez un objet. Lorsque vous spécifiez une somme de contrôle, elle est stockée avec l'objet et peut être validée lors du téléchargement de l'objet.

Les checksums fournissent une couche supplémentaire d'intégrité des données lorsque vous transférez des fichiers. Avec les checksums, vous pouvez vérifier la cohérence des données en confirmant que le fichier reçu correspond au fichier d'origine. Pour plus d'informations sur les checksums avec Amazon S3, consultez le guide de l'utilisateur d'Amazon Simple Storage Service, y compris les algorithmes pris en charge.

Vous avez la possibilité de choisir l'algorithme qui répond le mieux à vos besoins et de laisser le SDK calculer le checksum. Vous pouvez également fournir une valeur de somme de contrôle précalculée à l'aide de l'un des algorithmes pris en charge.

> **Note**
>
> À partir de la version 3.337.0 du AWS SDK pour PHP, le SDK fournit une protection d'intégrité par défaut en calculant automatiquement une somme de `CRC32` contrôle pour les téléchargements. Le SDK calcule cette somme de contrôle si vous ne fournissez pas de valeur de somme de contrôle précalculée ou si vous ne spécifiez pas d'algorithme que le SDK doit utiliser pour calculer une somme de contrôle.

Le SDK fournit également des paramètres globaux pour les protections de l'intégrité des données que vous pouvez définir en externe, que vous pouvez consulter dans le Guide de référence AWS des SDK et des outils.

Important

Pour utiliser l'`CRC32C`algorithme, votre environnement PHP nécessite l'installation de l'extension AWS Common Runtime (AWS CRT).

Nous discutons des sommes de contrôle en deux phases de demande : le téléchargement d'un objet et le téléchargement d'un objet.

Charger un objet

Vous chargez des objets sur Amazon S3 à l'aide de la méthode putObject du. `S3Client` Utilisez la `ChecksumAlgorithm` paire dans le tableau de paramètres pour activer le calcul de la somme de contrôle et spécifier l'algorithme.

```
$client = new \Aws\S3\S3Client(['region' => 'us-east-2']); // See the note below.
$result = $client->putObject([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'key',
        'ChecksumAlgorithm' => 'CRC32',
        'Body' => 'Object contents to test the checksum.'
]);
```

Si vous ne fournissez pas d'algorithme de somme de contrôle avec la demande, le comportement de somme de contrôle varie en fonction de la version du SDK que vous utilisez, comme indiqué dans le tableau suivant.

Comportement de somme de contrôle lorsqu'aucun algorithme de somme de contrôle n'est fourni

Version du SDK PHP	Comportement de Checksum
antérieur à 3.337.0	Le SDK ne calcule pas automatiquement une CRC-based somme de contrôle et ne la fournit pas dans la demande.

Version du SDK PHP	Comportement de Checksum
3.337.0 ou version ultérieure	Le SDK utilise l'`CRC32`algorithme pour calculer le checksum et le fournit dans la demande. Amazon S3 valide l'intégrité du transfert en calculant sa propre somme de `CRC32` contrôle et en la comparant à la somme de contrôle fournie par le SDK. Si les sommes de contrôle correspondent, la somme de contrôle est enregistrée avec l'objet.

Utiliser une valeur de somme de contrôle précalculée

Une valeur de somme de contrôle précalculée fournie avec la demande désactive le calcul automatique par le SDK et utilise la valeur fournie à la place.

L'exemple suivant montre une demande avec une somme de contrôle SHA256 précalculée.

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

$client = new S3Client([
    'region' => 'us-east-1',
]);

// Calculate the SHA256 checksum of the contents to be uploaded.
$contents = 'Object contents to test the checksum.';
$body = Psr7\Utils::streamFor($contents);
$sha256 = base64_encode(Psr7\Utils::hash($body, 'sha256', true));

$result = $client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'key',
    'Body' => $body,
    'ChecksumSHA256' => $sha256
]);
```

Si Amazon S3 détermine que la valeur de la somme de contrôle est incorrecte pour l'algorithme spécifié, le service renvoie une réponse d'erreur.

Chargements partitionnés

Vous pouvez également utiliser des checksums pour les téléchargements partitionnés.

Comme indiqué dans l'exemple suivant, spécifiez l'algorithme checksum sous la forme d'une paire clé-valeur dans le `params` tableau du constructeur. `MultipartUploader`

```
$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$stream = fopen("/path/to/large/file", "r");

$mpUploader = new MultipartUploader($s3Client, $stream, [
    'bucket' => 'amzn-s3-demo-bucket',
    'key' => 'key',
    'params' => ['ChecksumAlgorithm' => 'CRC32']
]);
```

Télécharger un objet

Lorsque vous utilisez la méthode GetObject pour télécharger un objet, le SDK valide automatiquement le checksum lorsque la valeur de la clé est. `ChecksumMode enabled`

La demande contenue dans l'extrait suivant demande au SDK de valider la somme de contrôle dans la réponse en calculant la somme de contrôle et en comparant les valeurs.

```
$result = $client->getObject([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'test-checksum-key',
        'ChecksumMode' => 'enabled',
]);
```

> **Note**
>
> Si l'objet n'a pas été chargé avec une somme de contrôle, aucune validation n'a lieu.

Exemples de code avec conseils pour AWS SDK pour PHP

Cette section contient des exemples de code illustrant les AWS scénarios courants utilisant le AWS SDK pour PHP.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Rubriques

- CloudFront Exemples Amazon utilisant la AWS SDK pour PHP version 3
- Signature de demandes de CloudSearch domaine Amazon personnalisées avec AWS SDK pour PHP la version 3
- CloudWatch Exemples Amazon utilisant la AWS SDK pour PHP version 3
- EC2 Exemples Amazon utilisant la AWS SDK pour PHP version 3
- Signature d'une demande OpenSearch de recherche Amazon Service avec AWS SDK pour PHP la version 3
- Gestion des identités et des accès AWS exemples utilisant la AWS SDK pour PHP version 3
- AWS Key Management Service exemples utilisant la AWS SDK pour PHP version 3
- Exemples d'Amazon Kinesis utilisant la version 3 AWS SDK pour PHP
- AWS Elemental MediaConvert exemples utilisant la AWS SDK pour PHP version 3
- Exemples d'Amazon S3 utilisant la AWS SDK pour PHP version 3
- Gestion des secrets à l'aide de l'API Secrets Manager et de la AWS SDK pour PHP version 3
- Exemples d'Amazon SES utilisant la AWS SDK pour PHP version 3
- Exemples d'Amazon SNS utilisant la version 3 AWS SDK pour PHP
- Exemples d'Amazon SQS utilisant la version 3 AWS SDK pour PHP
- Envoyez des événements aux terminaux EventBridge mondiaux Amazon à l'aide du AWS SDK pour PHP Version 3

CloudFront Exemples Amazon utilisant la AWS SDK pour PHP version 3

Amazon CloudFront est un service AWS Web qui accélère la diffusion de contenu Web statique et dynamique à partir de votre propre serveur Web ou d'un AWS serveur, tel qu'Amazon S3. CloudFront diffuse du contenu par le biais d'un réseau mondial de centres de données appelés emplacements périphériques. Lorsqu'un utilisateur demande du contenu que vous distribuez CloudFront, il est acheminé vers l'emplacement périphérique offrant la latence la plus faible. Si le contenu n'y est pas déjà mis en cache, CloudFront récupère une copie à partir du serveur d'origine, la remet, puis la met en cache pour les futures demandes.

Pour plus d'informations CloudFront, consultez le manuel Amazon CloudFront Developer Guide.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Gestion des CloudFront distributions Amazon à l'aide de CloudFront l'API et de la AWS SDK pour PHP version 3

Amazon met en CloudFront cache le contenu dans des emplacements périphériques du monde entier afin d'accélérer la distribution des fichiers statiques et dynamiques que vous stockez sur votre propre serveur ou sur un service Amazon tel qu'Amazon S3 et Amazon EC2. Lorsque les utilisateurs demandent du contenu à votre site Web, il le CloudFront diffuse à partir de l'emplacement périphérique le plus proche, si le fichier y est mis en cache. Sinon, CloudFront récupère une copie du fichier, la sert, puis la met en cache pour la prochaine demande. La mise en cache du contenu dans un emplacement périphérique permet de réduire la latence des demandes similaires des utilisateurs de cette zone.

Pour chaque CloudFront distribution que vous créez, vous spécifiez où se trouve le contenu et comment le distribuer lorsque les utilisateurs font des demandes. Cette rubrique porte sur les distributions pour les fichiers statiques et dynamiques tels que les fichiers HTML, CSS, JSON et d'image. Pour plus d'informations sur l'utilisation CloudFront de la vidéo à la demande, voir Vidéo à la demande et diffusion en direct avec CloudFront.

Les exemples suivants montrent comment :

- Créez une distribution à l'aide de CreateDistribution.
- Obtenez une distribution en utilisant GetDistribution.
- Répertoriez les distributions en utilisant ListDistributions.
- Mettez à jour les distributions en utilisant UpdateDistributions.
- Désactivez les distributions à l'aide de DisableDistribution.

- Supprimez les distributions à l'aide de DeleteDistributions.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon CloudFront, consultez le manuel du CloudFront développeur Amazon.

Création d'une CloudFront distribution

Créez une distribution à partir d'un compartiment Amazon S3. Dans l'exemple suivant, les paramètres facultatifs sont mis en commentaire, mais les valeurs par défaut sont affichées. Pour ajouter des personnalisations à votre distribution, supprimez la mise en commentaire de la valeur et du paramètre à l'intérieur de `$distribution`.

Pour créer une CloudFront distribution, utilisez l'CreateDistributionopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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 = 'amzn-s3-demo-bucket.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();
```

Récupérer une CloudFront distribution

Pour récupérer le statut et les détails d'une CloudFront distribution spécifiée, utilisez l'GetDistributionopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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 Répartition des listes

Obtenez une liste des CloudFront distributions existantes dans la AWS région spécifiée à partir de votre compte courant à l'aide de cette ListDistributionsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Mettre à jour une CloudFront distribution

La mise à jour d'une CloudFront distribution est similaire à la création d'une distribution. Toutefois, lorsque vous mettez à jour une distribution, davantage de champs sont obligatoires et toutes les valeurs doivent être incluses. Pour apporter des modifications à une distribution, nous vous recommandons tout d'abord de la récupérer, puis de mettre à jour les valeurs que vous souhaitez modifier dans le tableau `$distribution`.

Pour mettre à jour une CloudFront distribution spécifiée, utilisez l'UpdateDistributionopération.

Importations

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Désactiver une CloudFront distribution

Pour désactiver ou supprimer une distribution, faites passer son état de la valeur « deployed » (déployée) à la valeur « disabled » (désactivée).

Pour désactiver la CloudFront distribution spécifiée, utilisez l'DisableDistributionopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Supprimer une CloudFront distribution

Une fois qu'une distribution est à l'état désactivé, vous pouvez la supprimer.

Pour supprimer une CloudFront distribution spécifiée, utilisez l'DeleteDistributionopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Gestion des CloudFront invalidations Amazon à l'aide de l' CloudFront API et de la version 3 AWS SDK pour PHP

Amazon met en CloudFront cache des copies de fichiers statiques et dynamiques dans des emplacements périphériques du monde entier. Pour supprimer ou mettre à jour un fichier dans tous les emplacements périphériques, créez une invalidation pour chaque fichier ou pour un groupe de fichiers.

Chaque mois calendaire, vos 1 000 premières invalidations sont gratuites. Pour en savoir plus sur la suppression de contenu d'un emplacement CloudFront périphérique, consultez la section Invalidation de fichiers.

Les exemples suivants montrent comment :

- Créez une invalidation de distribution à l'aide CreateInvalidationde.
- Obtenez une invalidation de distribution en utilisant GetInvalidation.
- Répertoriez les distributions en utilisant ListInvalidations.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Pour plus d'informations sur l'utilisation d'Amazon CloudFront, consultez le manuel du CloudFront développeur Amazon.

Création d'une invalidation de distribution

Créez une invalidation de CloudFront distribution en spécifiant l'emplacement du chemin des fichiers que vous devez supprimer. Cet exemple invalide tous les fichiers de la distribution, mais vous pouvez identifier des fichiers spécifiques sous `Items`.

Pour créer une invalidation de CloudFront distribution, utilisez l'CreateInvalidationopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Obtenir l'invalidation d'une distribution

Pour récupérer le statut et les détails relatifs à l'invalidation d'une CloudFront distribution, utilisez l'GetInvalidationopération.

Importations

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

```
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Invalidations de distribution de listes

Pour répertorier toutes les invalidations CloudFront de distribution en cours, utilisez l'ListInvalidationsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Signature d'Amazon CloudFront URLs avec AWS SDK pour PHP la version 3

URLs Signed vous permet de permettre aux utilisateurs d'accéder à votre contenu privé. Une URL signée inclut des informations supplémentaires (par exemple, une heure d'expiration) qui vous permettent de mieux contrôler l'accès à votre contenu. Ces informations supplémentaires apparaissent dans une déclaration de politique, basée sur une politique prédéfinie ou une politique personnalisée. Pour plus d'informations sur la façon de configurer des distributions privées et sur les raisons pour lesquelles vous devez signer URLs, consultez la section Diffuser du contenu privé via Amazon CloudFront dans le manuel du CloudFront développeur Amazon.

- Créez une CloudFront URL Amazon signée à l'aide de GetSignedURL.
- Créez un CloudFront cookie Amazon signé à l'aide de getSignedCookie.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon CloudFront, consultez le manuel du CloudFront développeur Amazon.

Signature CloudFront URLs pour des distributions privées

Vous pouvez signer une URL à l'aide du CloudFront client dans le SDK. Tout d'abord, vous devez créer un objet `CloudFrontClient`. Vous pouvez signer CloudFront l'URL d'une ressource vidéo à l'aide d'une politique prédéfinie ou personnalisée.

Importations

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Utiliser une politique personnalisée lors de la création CloudFront URLs

Pour utiliser une stratégie personnalisée, fournissez la clé `policy` à la place de `expires`.

Importations

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Utiliser une URL CloudFront signée

La forme de l'URL signée diffère selon que l'URL que vous signez utilise le schéma « HTTP » ou « RTMP ». Dans le cas du schéma « HTTP », l'intégralité de l'URL absolue est renvoyée. Dans le cas du schéma « RTMP », seule l'URL relative est renvoyée pour simplifier le processus. Cela est dû au fait que certains lecteurs exigent que l'hôte et le chemin soient fournis en tant que paramètres séparés.

L'exemple suivant montre comment vous pouvez utiliser l'URL signée pour créer une page Web qui affiche une vidéo en utilisant JWPlayer. Le même type de technique s'appliquerait aux autres joueurs, par exemple FlowPlayer, mais nécessiterait un code différent côté client.

```
<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>
```

CloudFront Cookies de signature pour les distributions privées

Comme alternative aux cookies signés URLs, vous pouvez également autoriser les clients à accéder à une distribution privée via des cookies signés. Les cookies signés vous permettent de fournir l'accès à plusieurs fichiers restreints : par exemple, tous les fichiers d'une vidéo au format HLS ou tous les fichiers de la section des abonnés d'un site web. Pour plus d'informations sur les raisons pour lesquelles vous souhaiterez peut-être utiliser des cookies signés au lieu de cookies signés URLs (ou vice versa), consultez Choisir entre des cookies signés URLs et des cookies signés dans le manuel Amazon CloudFront Developer Guide.

La création d'un cookie signé est similaire à celle d'une URL signée. La seule différence est la méthode appelée (`getSignedCookie` au lieu de `getSignedUrl`).

Importations

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Utiliser une politique personnalisée lors de la création de CloudFront cookies

Comme avec `getSignedUrl`, vous pouvez fournir un paramètre `'policy'` à la place d'un paramètre `expires` et d'un paramètre `url` pour signer un cookie avec une stratégie personnalisée. Une stratégie personnalisée peut contenir des caractères génériques dans la ressource clé. Cela vous permet de créer un seul cookie signé pour plusieurs fichiers.

`getSignedCookie` renvoie un tableau de paires clé-valeur. Toutes ces paires doivent être définies comme des cookies pour accorder l'accès à une distribution privée.

Importations

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Envoyer CloudFront des cookies au client Guzzle

Vous pouvez également transmettre ces cookies à un `GuzzleHttp\Cookie\CookieJar` pour une utilisation avec un client Guzzle.

```
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');
```

Pour plus d'informations, consultez la section Utilisation de cookies signés dans le manuel Amazon CloudFront Developer Guide.

Signature de demandes de CloudSearch domaine Amazon personnalisées avec AWS SDK pour PHP la version 3

Les demandes CloudSearch de domaine Amazon peuvent être personnalisées au-delà de ce qui est pris en charge par le AWS SDK pour PHP. Dans les cas où vous devez envoyer des demandes personnalisées à des domaines protégés par l'authentification IAM, vous pouvez utiliser les fournisseurs d'informations d'identification et les signataires du SDK pour signer toute demande PSR-7.

Par exemple, si vous suivez le Guide de démarrage Cloud Search et que vous souhaitez utiliser un domaine protégé par IAM pour l'étape 3, vous devez signer et exécuter votre demande comme suit.

Les exemples suivants montrent comment :

- Signez une demande avec le protocole de AWS signature à l'aide de SignatureV4.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Signer la demande CloudSearch de domaine Amazon

Importations

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

```
use Aws\Credentials\CredentialProvider;
use Aws\Signature\SignatureV4;
use GuzzleHttp\Client;
use GuzzleHttp\Psr7\Request;
```

Exemple de code

```
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();
```

CloudWatch Exemples Amazon utilisant la AWS SDK pour PHP version 3

Amazon CloudWatch (CloudWatch) est un service Web qui surveille vos ressources Amazon Web Services et les applications que vous exécutez AWS en temps réel. Vous pouvez les utiliser CloudWatch pour collecter et suivre les métriques, qui sont des variables que vous pouvez mesurer pour vos ressources et vos applications. CloudWatch les alarmes envoient des notifications ou modifient automatiquement les ressources que vous surveillez en fonction des règles que vous définissez.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Rubriques

- Utilisation des CloudWatch alarmes Amazon avec AWS SDK pour PHP la version 3
- Obtenir des statistiques auprès d'Amazon CloudWatch avec AWS SDK pour PHP la version 3
- Publication de métriques personnalisées dans Amazon CloudWatch avec AWS SDK pour PHP la version 3
- Envoyer des événements à Amazon CloudWatch Events avec AWS SDK pour PHP la version 3
- Utilisation d'actions d'alarme avec les CloudWatch alarmes Amazon avec AWS SDK pour PHP la version 3

Utilisation des CloudWatch alarmes Amazon avec AWS SDK pour PHP la version 3

Une CloudWatch alarme Amazon surveille une seule métrique pendant une période que vous spécifiez. Elle réalise une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes.

Les exemples suivants montrent comment :

- Décrivez une alarme en utilisant DescribeAlarms.
- Créez une alarme en utilisant PutMetricAlarm.
- Supprimer une alarme à l'aide de DeleteAlarms.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Description des alarmes

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Créer une alarme

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Supprimer des alertes

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Obtenir des statistiques auprès d'Amazon CloudWatch avec AWS SDK pour PHP la version 3

Les métriques sont des données sur les performances de vos systèmes. Vous pouvez activer le suivi détaillé de certaines ressources, telles que vos EC2 instances Amazon, ou de vos propres indicateurs d'application.

Les exemples suivants montrent comment :

- Répertoriez les métriques en utilisant ListMetrics.
- Récupérez les alarmes d'une métrique à l'aide de DescribeAlarmsForMetric.
- Obtenez des statistiques pour une métrique spécifiée à l'aide de GetMetricStatistics.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Répertorier les métriques

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Récupérer les alarmes pour une métrique

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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' => 'amzn-s3-demo-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();
```

Obtenir des statistiques de métriques

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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' => 'amzn-s3-demo-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();
```

Publication de métriques personnalisées dans Amazon CloudWatch avec AWS SDK pour PHP la version 3

Les métriques sont des données sur les performances de vos systèmes. Une alarme surveille une métrique sur la période que vous spécifiez. Elle réalise une ou plusieurs actions en fonction de la valeur de la métrique, par rapport à un seuil donné sur un certain nombre de périodes.

Les exemples suivants montrent comment :

- Publiez des données métriques à l'aide de PutMetricData.

- Créez une alarme en utilisant PutMetricAlarm.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Publier des données métriques

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Créer une alarme

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Envoyer des événements à Amazon CloudWatch Events avec AWS SDK pour PHP la version 3

CloudWatch Events fournit un flux en temps quasi réel d'événements système décrivant les modifications apportées aux ressources Amazon Web Services (AWS) à différentes cibles. À l'aide de règles simples, vous pouvez faire correspondre les événements et les acheminer vers un ou plusieurs flux ou fonctions cibles.

Les exemples suivants montrent comment :

- Créez une règle à l'aide de PutRule.
- Ajoutez des cibles à une règle à l'aide de PutTargets.
- Envoyez des événements personnalisés à CloudWatch Events à l'aide de PutEvents.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Création d'une règle

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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());
}
```

Ajouter des cibles à une règle

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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());
}
```

Envoyer des événements personnalisés

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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());
}
```

Utilisation d'actions d'alarme avec les CloudWatch alarmes Amazon avec AWS SDK pour PHP la version 3

Utilisez des actions d'alarme pour créer des alarmes qui arrêtent, mettent fin, redémarrent ou restaurent automatiquement vos EC2 instances Amazon. Vous pouvez utiliser les actions d'arrêt ou de mise hors service quand vous n'avez plus besoin qu'une instance s'exécute. Vous pouvez utiliser les actions de redémarrage et de récupération pour redémarrer automatiquement ces instances.

Les exemples suivants montrent comment :

- Activez les actions pour les alarmes spécifiées à l'aide de EnableAlarmActions.
- Désactivez les actions pour les alarmes spécifiées à l'aide de DisableAlarmActions.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Activer des actions d'alerte

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

Désactiver des actions d'alerte

Importations

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
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();
```

EC2 Exemples Amazon utilisant la AWS SDK pour PHP version 3

Amazon Elastic Compute Cloud (Amazon EC2) est un service Web qui fournit un hébergement de serveurs virtuels dans le cloud. Il est conçu pour faciliter le cloud computing à l'échelle du Web pour les développeurs en fournissant une capacité de calcul redimensionnable.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Rubriques

- Gestion des EC2 instances Amazon à l'aide de la AWS SDK pour PHP version 3
- Utilisation d'adresses IP élastiques avec Amazon EC2 version 3 AWS SDK pour PHP
- Utilisation des régions et des zones de disponibilité pour Amazon EC2 avec AWS SDK pour PHP la version 3
- Utilisation des paires de EC2 clés Amazon avec AWS SDK pour PHP la version 3
- Utilisation des groupes de sécurité dans Amazon EC2 avec AWS SDK pour PHP la version 3

Gestion des EC2 instances Amazon à l'aide de la AWS SDK pour PHP version 3

Les exemples suivants montrent comment :

- Décrivez EC2 les instances Amazon utilisant DescribeInstances.
- Activez la surveillance détaillée d'une instance en cours d'exécution à l'aide de MonitorInstances.
- Désactivez la surveillance d'une instance en cours d'exécution à l'aide de UnmonitorInstances.
- Démarrez une AMI basée sur Amazon EBS que vous avez précédemment arrêtée d'utiliser. StartInstances
- Arrêtez l'utilisation d'une instance basée sur Amazon EBS. StopInstances
- Demandez le redémarrage d'une ou de plusieurs instances à l'aide de RebootInstances.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Décrire des instances

Importations

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

use Aws\Ec2\Ec2Client;
```

Exemple de code

```
$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";
    }
}
```

Activer et désactiver la surveillance

Importations

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

Exemple de code

```
$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);
```

lancer et arrêter une instance ;

Importations

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

Exemple de code

```
$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);
```

Redémarrer une instance

Importations

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

Exemple de code

```
$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);
```

Utilisation d'adresses IP élastiques avec Amazon EC2 version 3 AWS SDK pour PHP

Une EIP est une adresse IP statique conçue pour le cloud computing. Une adresse IP élastique est associée à votre Compte AWS. Il s'agit d'une adresse IP publique accessible depuis Internet. Si votre instance ne dispose pas d'une adresse IP publique, vous pouvez lui associer une adresse IP Elastic pour établir la communication avec Internet.

Les exemples suivants montrent comment :

- Décrivez une ou plusieurs de vos instances en utilisant DescribeInstances.
- Obtenez une adresse IP élastique à l'aide de AllocateAddress.
- Associez une adresse IP élastique à une instance à l'aide de AssociateAddress.
- Libérez une adresse IP élastique à l'aide de ReleaseAddress.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Décrire une instance

Importations

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

use Aws\Ec2\Ec2Client;
```

Exemple de code

```
$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";
    }
}
```

Attribuer et associer une adresse

Importations

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

Exemple de code

```
$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);
```

Publier une adresse

Importations

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

Exemple de code

```
$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);
```

Utilisation des régions et des zones de disponibilité pour Amazon EC2 avec AWS SDK pour PHP la version 3

Amazon EC2 est hébergé sur plusieurs sites dans le monde entier. Ces emplacements sont composés de AWS régions et de zones de disponibilité. Chaque région est une zone géographique distincte, avec plusieurs emplacements isolés appelés zones de disponibilité. Amazon EC2 permet de placer des instances et des données sur plusieurs sites.

Les exemples suivants montrent comment :

- Décrivez les zones de disponibilité que vous pouvez utiliser DescribeAvailabilityZones.
- Décrivez les AWS régions que vous pouvez actuellement utiliser DescribeRegions.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Décrire les zones de disponibilité

Importations

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

Exemple de code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

Décrire les régions

Importations

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

Exemple de code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

Utilisation des paires de EC2 clés Amazon avec AWS SDK pour PHP la version 3

Amazon EC2 utilise le chiffrement à clé publique pour chiffrer et déchiffrer les informations de connexion. Le chiffrement de clé publique utilise une clé publique pour chiffrer les données. Ensuite, le destinataire utilise la clé privée pour déchiffrer les données. La clé publique et la clé privée constituent une paire de clés.

Les exemples suivants montrent comment :

- Créez une paire de clés RSA 2048 bits à l'aide de. CreateKeyPair
- Supprimez une paire de clés spécifiée à l'aide de DeleteKeyPair.
- Décrivez une ou plusieurs de vos paires de clés en utilisant DescribeKeyPairs.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Création d'une paire de clés

Importations

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

Exemple de code

```
$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);
```

Supprimer une paire de clés

Importations

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

Exemple de code

```
$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);
```

Décrire des paires de clés

Importations

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

Exemple de code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

Utilisation des groupes de sécurité dans Amazon EC2 avec AWS SDK pour PHP la version 3

Un groupe EC2 de sécurité Amazon agit comme un pare-feu virtuel qui contrôle le trafic d'une ou de plusieurs instances. Vous ajoutez des règles à chaque groupe de sécurité pour autoriser le trafic vers ou depuis ses instances associées. Vous pouvez modifier les règles pour un groupe de sécurité à

la fois. Les nouvelles règles sont automatiquement appliquées à toutes les instances associées au groupe de sécurité.

Les exemples suivants montrent comment :

- Décrivez un ou plusieurs de vos groupes de sécurité en utilisant DescribeSecurityGroups.
- Ajoutez une règle d'entrée à un groupe de sécurité à l'aide AuthorizeSecurityGroupIngressde.
- Créez un groupe de sécurité à l'aide de CreateSecurityGroup.
- Supprimez un groupe de sécurité à l'aide de DeleteSecurityGroup.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Décrire des groupes de sécurité

Importations

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

Exemple de code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

Ajouter une règle d'entrée

Importations

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

Exemple de code

```
$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);
```

Création d’un groupe de sécurité

Importations

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

Exemple de code

```
$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';
```

Supprimer un groupe de sécurité

Importations

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

Exemple de code

```
$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);
```

Signature d'une demande OpenSearch de recherche Amazon Service avec AWS SDK pour PHP la version 3

Amazon OpenSearch Service est un service géré qui facilite le déploiement, l'exploitation et le dimensionnement d'Amazon OpenSearch Service, un moteur de recherche et d'analyse open source

populaire. OpenSearch Le service offre un accès direct à l'API Amazon OpenSearch Service. Cela signifie que les développeurs peuvent utiliser les outils qu'ils connaissent bien, ainsi que des options de sécurité robustes. De nombreux clients Amazon OpenSearch Service prennent en charge la signature des demandes, mais si vous utilisez un client qui ne le fait pas, vous pouvez signer des demandes PSR-7 arbitraires avec les fournisseurs d'informations d'identification et les signataires intégrés du. AWS SDK pour PHP

Les exemples suivants montrent comment :

- Signez une demande avec le protocole de AWS signature à l'aide de SignatureV4.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Signature d'une demande OpenSearch de service

OpenSearch Le service utilise la version 4 de Signature. Cela signifie que vous devez signer les demandes en fonction du nom de signature du service (`es`dans ce cas) et de la AWS région de votre domaine de OpenSearch service. La liste complète des régions prises en charge par le OpenSearch service est disponible sur la page AWS Régions et points de terminaison du Référence générale d'Amazon Web Services. Toutefois, dans cet exemple, nous signons des demandes relatives à un domaine de OpenSearch service de la `us-west-2` région.

Vous devez fournir des informations d'identification, ce que vous pouvez faire soit avec la chaîne de fournisseurs par défaut du SDK, soit avec toute forme d'informations d'identification décrite dans Informations d'identification pour la AWS SDK pour PHP version 3. Vous aurez également besoin d'une demande PSR-7 (censée s'appeler `$psr7Request` dans le code ci-dessous).

```
// 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);
```

Gestion des identités et des accès AWS exemples utilisant la AWS SDK pour PHP version 3

Gestion des identités et des accès AWS (IAM) est un service Web qui permet aux clients d'Amazon Web Services de gérer les utilisateurs et les autorisations des utilisateurs dans AWS. Le service est destiné aux organisations ayant plusieurs utilisateurs ou des systèmes dans le cloud qui utilisent AWS des produits. Avec IAM, vous pouvez gérer de manière centralisée les utilisateurs, les informations d'identification de sécurité telles que les clés d'accès et les autorisations qui contrôlent les AWS ressources auxquelles les utilisateurs peuvent accéder.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Rubriques

- Gestion des clés d'accès IAM avec AWS SDK pour PHP la version 3
- Gestion des utilisateurs IAM avec la AWS SDK pour PHP version 3
- Utilisation d'alias de compte IAM avec AWS SDK pour PHP la version 3
- Utilisation des politiques IAM avec la AWS SDK pour PHP version 3
- Utilisation de certificats de serveur IAM avec AWS SDK pour PHP la version 3

Gestion des clés d'accès IAM avec AWS SDK pour PHP la version 3

Les utilisateurs ont besoin de leurs propres clés d'accès pour effectuer des appels programmatiques. AWS Pour répondre à ce besoin, vous pouvez créer, modifier, afficher ou faire pivoter des clés d'accès (clé d'accès IDs et clés d'accès secrètes) pour les utilisateurs IAM. Par défaut, lorsque vous créez une clé d'accès, son état est Active. Cela signifie que l'utilisateur peut se servir de la clé d'accès pour effectuer des appels d'API.

Les exemples suivants montrent comment :

- Créez une clé d'accès secrète et l'ID de clé d'accès correspondant à l'aide de CreateAccessKey.
- Renvoie des informations sur la clé d'accès IDs associée à un utilisateur IAM utilisant ListAccessKeys.
- Récupérez des informations sur la date à laquelle une clé d'accès a été utilisée pour la dernière fois en utilisant GetAccessKeyLastUsed.
- Modifiez le statut d'une clé d'accès d'Actif à Inactif, ou vice versa, en utilisant UpdateAccessKey.
- Supprimez une paire de clés d'accès associée à un utilisateur IAM à l'aide DeleteAccessKeyde.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Créer une clé d'accès

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Répertorier des clés d'accès

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Obtenir des informations sur la dernière utilisation d'une clé d'accès

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Mettre à jour une clé d'accès

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer une clé d'accès

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Gestion des utilisateurs IAM avec la AWS SDK pour PHP version 3

Un utilisateur IAM est une entité que vous créez AWS pour représenter la personne ou le service avec AWS lequel il interagit. Un utilisateur se AWS compose d'un nom et d'informations d'identification.

Les exemples suivants montrent comment :

- Créez un nouvel utilisateur IAM à l'aide CreateUserde.
- Répertoriez les utilisateurs IAM en utilisant ListUsers.
- Mettez à jour un utilisateur IAM à l'aide UpdateUserde.
- Récupérez des informations sur un utilisateur IAM à l'aide GetUserde.
- Supprimez un utilisateur IAM à l'aide DeleteUserde.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Créer un utilisateur IAM

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Répertorier les utilisateurs IAM

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Mettre à jour un utilisateur IAM

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Obtenir des informations sur un utilisateur IAM

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer un utilisateur IAM

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Utilisation d'alias de compte IAM avec AWS SDK pour PHP la version 3

Si vous souhaitez que l'URL de votre page de connexion contienne le nom de votre entreprise ou un autre identifiant convivial au lieu de votre Compte AWS identifiant, vous pouvez créer un alias pour votre Compte AWS identifiant. Si vous créez un Compte AWS alias, l'URL de votre page de connexion change pour intégrer l'alias.

Les exemples suivants montrent comment :

- Créez un alias à l'aide de CreateAccountAlias.
- Répertoriez l'alias associé à l' Compte AWS utilisation ListAccountAliases.
- Supprimez un alias à l'aide de DeleteAccountAlias.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Créer un alias

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Répertorier des alias de compte

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer un alias

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Utilisation des politiques IAM avec la AWS SDK pour PHP version 3

Vous devez accorder des autorisations à un utilisateur en créant une stratégie. Une stratégie est un document qui répertorie les actions qu'un utilisateur peut effectuer ainsi que les ressources que ces actions peuvent affecter. Les actions ou ressources qui ne sont pas explicitement autorisées sont refusées par défaut. Vous pouvez créer des stratégies et les attacher à des utilisateurs, à des groupes d'utilisateurs, à des rôles pris en charge par des utilisateurs et à des ressources.

Les exemples suivants montrent comment :

- Créez une politique gérée à l'aide de CreatePolicy.
- Attachez une politique à un rôle à l'aide de AttachRolePolicy.
- Attachez une politique à un utilisateur en utilisant AttachUserPolicy.
- Attachez une politique à un groupe à l'aide de AttachGroupPolicy.
- Supprimez une politique de rôle à l'aide de DetachRolePolicy.
- Supprimez une politique utilisateur à l'aide de DetachUserPolicy.
- Supprimez une politique de groupe à l'aide de DetachGroupPolicy.
- Supprimez une politique gérée à l'aide de DeletePolicy.
- Supprimez une politique de rôle à l'aide de DeleteRolePolicy.
- Supprimez une politique utilisateur à l'aide de DeleteUserPolicy.
- Supprimez une politique de groupe à l'aide de DeleteGroupPolicy.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Créer une politique

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Attacher une politique à un rôle

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Attacher une politique à un utilisateur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Associer une politique à un groupe

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Détacher une politique utilisateur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Détacher une politique de groupe

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer une politique

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer une politique de rôle

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer une politique utilisateur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
```

```
}
```

Supprimer une politique de groupe

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Utilisation de certificats de serveur IAM avec AWS SDK pour PHP la version 3

Pour activer les connexions HTTPS à votre site Web ou à votre application AWS, vous avez besoin d'un certificat de SSL/TLS serveur. Pour utiliser un certificat que vous avez obtenu auprès d'un fournisseur externe avec votre site Web ou votre application AWS, vous devez télécharger le certificat dans IAM ou l'importer dans AWS Certificate Manager.

Les exemples suivants montrent comment :

- Répertoriez les certificats stockés dans IAM à l'aide ListServerCertificatesde.
- Récupérez les informations relatives à un certificat à l'aide de GetServerCertificate.
- Mettez à jour un certificat à l'aide de UpdateServerCertificate.
- Supprimez un certificat à l'aide de DeleteServerCertificate.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Lister les certificats de serveur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Récupérer un certificat de serveur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Mettre à jour un certificat de serveur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

```
$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());
}
```

Supprimer un certificat de serveur

Importations

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

Exemple de code

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

```
try {
    $result = $client->deleteServerCertificate([
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS Key Management Service exemples utilisant la AWS SDK pour PHP version 3

AWS Key Management Service (AWS KMS) est un service géré qui vous permet de créer et de contrôler facilement les clés de chiffrement utilisées pour chiffrer vos données. Pour plus d'informations AWS KMS, consultez la documentation Amazon KMS. Que vous écriviez des applications PHP sécurisées ou que vous envoyiez des données à d'autres AWS services, AWS KMS cela vous permet de contrôler qui peut utiliser vos clés et accéder à vos données cryptées.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Rubriques

- Utilisation des clés à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3
- Chiffrement et déchiffrement des clés de AWS KMS données à l'aide de la version 3 AWS SDK pour PHP
- Utilisation des politiques AWS KMS clés à l'aide de la AWS SDK pour PHP version 3
- Utilisation des subventions à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3
- Utilisation d'alias à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3

Utilisation des clés à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3

Les principales ressources contenues dans AWS Key Management Service (AWS KMS) sont AWS KMS keys. Vous pouvez utiliser une clé KMS pour chiffrer vos données.

Les exemples suivants montrent comment :

- Créez une clé KMS client à l'aide de CreateKey.

- Générez une clé de données à l'aide de GenerateDataKey.
- Affichez une clé KMS à l'aide de DescribeKey.
- Obtenez les clés IDs et les ARNS des clés KMS à l'aide ListKeysde.
- Activez les clés KMS à l'aide de EnableKey.
- Désactivez les clés KMS à l'aide de DisableKey.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation de AWS Key Management Service (AWS KMS), consultez le manuel du AWS KMS développeur.

Création d'une clé KMS

Pour créer une clé KMS, utilisez l'CreateKeyopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Générer une clé de données

Pour générer une clé de chiffrement des données, utilisez l'GenerateDataKeyopération. Cette opération renvoie des copies en texte brut et chiffrées de la clé de données créée. Spécifiez la clé de données AWS KMS key sous laquelle vous souhaitez générer la clé de données.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Afficher une clé KMS

Pour obtenir des informations détaillées sur une clé KMS, notamment le nom de ressource Amazon (ARN) et l'état de la clé KMS, utilisez l'DescribeKeyopération.

`DescribeKey` ne récupère pas les alias. Pour obtenir des alias, utilisez l'ListAliasesopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Obtenez l'ID de clé et ARNs la clé d'une clé KMS

Pour obtenir l'ID et l'ARN de la clé KMS, utilisez l'ListAliasesopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Activer une clé KMS

Pour activer une clé KMS désactivée, utilisez l'EnableKeyopération.

Importations

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

```
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Désactiver une clé KMS

Pour désactiver une clé KMS, utilisez l'DisableKeyopération. La désactivation d'une clé KMS empêche son utilisation.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Chiffrement et déchiffrement des clés de AWS KMS données à l'aide de la version 3 AWS SDK pour PHP

Les clés de données sont des clés de chiffrement que vous pouvez utiliser pour chiffrer des données, y compris de grandes quantités de données et d'autres clés de chiffrement des données.

Vous pouvez utiliser AWS Key Management Service an's (AWS KMS) AWS KMS keypour générer, chiffrer et déchiffrer des clés de données.

Les exemples suivants montrent comment :

- Chiffrer une clé de données à l'aide d'Encrypt.
- Déchiffrer une clé de données à l'aide de Decrypt.
- Rechiffrez une clé de données avec une nouvelle clé KMS à l'aide de. ReEncrypt

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation de AWS Key Management Service (AWS KMS), consultez le manuel du AWS KMS développeur.

Encrypt

L'opération Encrypt est conçue pour chiffrer des clés de données, mais elle n'est pas fréquemment utilisée. Les GenerateDataKeyWithoutPlaintextopérations GenerateDataKeyet renvoient des clés de données chiffrées. Vous pouvez utiliser `Encypt` cette méthode lorsque vous déplacez des données chiffrées vers une nouvelle AWS région et que vous souhaitez chiffrer leur clé de données à l'aide d'une clé KMS dans la nouvelle région.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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

Pour déchiffrer une clé de données, utilisez l'opération Decrypt.

La valeur `ciphertextBlob` que vous spécifiez doit être la valeur du `CiphertextBlob` champ provenant d'une réponse GenerateDataKeyGenerateDataKeyWithoutPlaintext, ou Encrypt.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Rechiffrer

Pour déchiffrer une clé de données chiffrée, puis la rechiffrer immédiatement sous une autre clé KMS, utilisez l'opération. ReEncrypt Les opérations sont entièrement effectuées du côté serveur à l'intérieur AWS KMS, de sorte qu'elles n'exposent jamais votre texte brut en dehors de AWS KMS.

La valeur `ciphertextBlob` que vous spécifiez doit être la valeur du `CiphertextBlob` champ provenant d'une réponse GenerateDataKeyGenerateDataKeyWithoutPlaintext, ou Encrypt.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Utilisation des politiques AWS KMS clés à l'aide de la AWS SDK pour PHP version 3

Lorsque vous créez une clé KMS AWS KMS key, vous déterminez qui peut utiliser et gérer cette clé KMS. Ces autorisations sont contenues dans un document appelé politique de clé. Vous pouvez utiliser la politique clé pour ajouter, supprimer ou modifier des autorisations à tout moment pour une clé KMS gérée par le client, mais vous ne pouvez pas modifier la politique clé pour une clé KMS AWS gérée. Pour plus d'informations, consultez Authentification et contrôle d'accès pour AWS KMS.

Les exemples suivants montrent comment :

- Répertoriez les noms des principales politiques à l'aide de ListKeyPolicies.
- Obtenez une politique clé en utilisant GetKeyPolicy.
- Définissez une politique clé à l'aide de PutKeyPolicy.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation de AWS Key Management Service (AWS KMS), consultez le manuel du AWS KMS développeur.

Répertorier toutes les politiques clés

Pour obtenir les noms des politiques clés pour une clé KMS, utilisez l'`ListKeyPolicies`opération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Récupérer une politique clé

Pour obtenir la politique de clé d'une clé KMS, utilisez l'`GetKeyPolicy`opération.

`GetKeyPolicy` nécessite un nom de politique. À moins que vous n'ayez créé une politique clé lors de la création de la clé KMS, le seul nom de politique valide est le nom par défaut. Pour en savoir plus sur la politique relative aux clés par défaut, consultez le guide du AWS Key Management Service développeur.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Définissez une politique clé

Pour établir ou modifier une politique de clé pour une clé KMS, utilisez l'`PutKeyPolicy`opération.

`PutKeyPolicy` nécessite un nom de politique. À moins que vous n'ayez créé une politique clé lors de la création de la clé KMS, le seul nom de politique valide est le nom par défaut. Pour en savoir plus sur la politique relative aux clés par défaut, consultez le guide du AWS Key Management Service développeur.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Utilisation des subventions à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3

Un octroi est un autre mécanisme permettant de fournir des autorisations. C'est une alternative à la politique clé. Vous pouvez utiliser des subventions pour accorder un accès à long terme qui permet AWS aux principaux d'utiliser votre AWS Key Management Service (AWS KMS) géré par le client AWS KMS keys. Pour plus d'informations, consultez la section Subventions AWS KMS dans le guide du AWS Key Management Service développeur.

Les exemples suivants montrent comment :

- Créez une autorisation pour une clé KMS à l'aide de CreateGrant.
- Affichez une autorisation pour une clé KMS à l'aide de ListGrants.
- Retirez une subvention pour une clé KMS en utilisant RetireGrant.
- Révoquez une autorisation pour une clé KMS en utilisant RevokeGrant.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation de AWS Key Management Service (AWS KMS), consultez le manuel du AWS KMS développeur.

Créer un octroi

Pour créer une subvention pour un AWS KMS key, utilisez l'CreateGrantopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Affichage d'un octroi

Pour obtenir des informations détaillées sur les subventions sur un AWS KMS key, utilisez l'ListGrantsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Retrait d'une subvention

Pour annuler une subvention pour un AWS KMS key, utilisez l'RetireGrantopération. Résilier un octroi pour nettoyer une fois que vous avez fini de l'utiliser.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Révoquer une subvention

Pour révoquer une subvention accordée à un AWS KMS key, utilisez l'RevokeGrantopération. Vous pouvez révoquer un octroi pour refuser explicitement les opérations qui en dépendent.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Utilisation d'alias à l'aide de l' AWS KMS API et de la AWS SDK pour PHP version 3

AWS Key Management Service (AWS KMS) fournit un nom d'affichage facultatif pour un alias AWS KMS keyappelé.

Les exemples suivants montrent comment :

- Créez un alias à l'aide de CreateAlias.
- Affichez un alias à l'aide de ListAliases.
- Mettez à jour un alias à l'aide de UpdateAlias.
- Supprimez un alias à l'aide de DeleteAlias.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Pour plus d'informations sur l'utilisation de AWS Key Management Service (AWS KMS), consultez le manuel du AWS KMS développeur.

Création d’un alias

Pour créer un alias pour une clé KMS, utilisez l'CreateAliasopération. L'alias doit être unique dans le compte et dans AWS la région. Si vous créez un alias pour une clé KMS qui possède déjà un alias, `CreateAlias` crée un autre alias pour la même clé KMS. Cette opération ne remplace pas l'alias existant.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Afficher un alias

Pour répertorier tous les alias figurant dans l'annonce de l' Compte AWS appelant Région AWS, utilisez l'ListAliasesopération.

Importations

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

```
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Mettre à jour un alias

Pour associer un alias existant à une autre clé KMS, utilisez l'UpdateAliasopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Suppression d’un alias

Pour supprimer un alias, utilisez l'DeleteAliasopération. La suppression d'un alias n'a aucun effet sur la clé KMS sous-jacente.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Exemples d'Amazon Kinesis utilisant la version 3 AWS SDK pour PHP

Amazon Kinesis est un AWS service qui collecte, traite et analyse les données en temps réel. Configurez vos flux de données avec Amazon Kinesis Data Streams ou utilisez Amazon Data Firehose pour envoyer des données vers Amazon S3, Service OpenSearch , Amazon Redshift ou Splunk.

Pour plus d'informations sur Kinesis, consultez la documentation Amazon Kinesis.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Rubriques

- Création de flux de données à l'aide de l'API Kinesis Data Streams et de AWS SDK pour PHP la version 3
- Gérez les fragments de données à l'aide de l'API Kinesis Data Streams et AWS SDK pour PHP de la version 3
- Création de flux de diffusion à l'aide de l'API Firehose et de la version 3 AWS SDK pour PHP

Création de flux de données à l'aide de l'API Kinesis Data Streams et de AWS SDK pour PHP la version 3

Amazon Kinesis Data Streams vous permet d'envoyer des données en temps réel. Créez un producteur de données avec Kinesis Data Streams qui fournit les données à la destination configurée chaque fois que vous ajoutez des données.

Pour plus d'informations, consultez la section Création et gestion de flux dans le manuel Amazon Kinesis Developer Guide.

Les exemples suivants montrent comment :

- Créez un flux de données à l'aide de CreateAlias.
- Obtenez des informations sur un seul flux de données à l'aide de DescribeStream.
- Répertoriez les flux de données existants en utilisant ListStreams.
- Envoyez des données vers un flux de données existant à l'aide de PutRecord.
- Supprimez un flux de données à l'aide de DeleteStream.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon Kinesis Developer Guide, consultez le Amazon Kinesis Data Streams Developer Guide.

Création d'un flux de données à l'aide d'un flux de données Kinesis

Établissez un flux de données Kinesis dans lequel vous pouvez envoyer des informations à traiter par Kinesis à l'aide de l'exemple de code suivant. Pour en savoir plus sur la création et la mise à jour de flux de données, consultez le guide du développeur Amazon Kinesis.

Pour créer un flux de données Kinesis, utilisez l'CreateStreamopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Récupérer un flux de données

Obtenez des détails sur un flux de données existant à l'aide de l'exemple de code suivant. Par défaut, cela renvoie des informations sur les 10 premières partitions connectées au flux de données Kinesis spécifié. N'oubliez pas `StreamStatus` de vérifier la réponse avant d'écrire des données dans un flux de données Kinesis.

Pour récupérer les informations relatives à un flux de données Kinesis spécifique, utilisez l'DescribeStreamopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier les flux de données existants connectés à Kinesis

Répertoriez les 10 premiers flux de données provenant Compte AWS de la AWS région sélectionnée. Utilisez le `HasMoreStreams retourné pour déterminer si plusieurs flux sont associés à votre compte.

Pour répertorier vos flux de données Kinesis, utilisez l'ListStreamsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Envoyer des données vers un flux de données existant

Une fois qu'un flux de données est créé, utilisez l'exemple suivant pour envoyer des données. Avant d'y envoyer des données, utilisez `DescribeStream` pour vérifier si les données `StreamStatus` sont actives.

Pour écrire un seul enregistrement de données dans un flux de données Kinesis, utilisez l'PutRecordopération. Pour écrire jusqu'à 500 enregistrements dans un flux de données Kinesis, utilisez l'PutRecordsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un flux de données

Cet exemple indique comment supprimer un flux de données. La suppression d'un flux de données entraîne également la suppression des données envoyées au flux de données. Les flux de données Active Kinesis passent à l'état DELETING jusqu'à ce que la suppression du flux soit terminée. Lorsqu'il est à l'état SUPPRESSION EN COURS, le flux continue à traiter des données.

Pour supprimer un flux de données Kinesis, utilisez l'DeleteStreamopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Gérez les fragments de données à l'aide de l'API Kinesis Data Streams et AWS SDK pour PHP de la version 3

Amazon Kinesis Data Streams vous permet d'envoyer des données en temps réel à un terminal. Le débit du flux de données varie en fonction du nombre de partitions dans votre flux.

Vous pouvez écrire 1 000 enregistrements par seconde sur une seule partition. Chaque partition dispose également d'une limite de chargement de 1 Mo par seconde. L'utilisation est calculée et facturée par partition, de manière à utiliser ces exemples pour gérer les données de capacité et de coût de votre flux.

Les exemples suivants montrent comment :

- Répertoriez les fragments d'un flux à l'aide ListShardsde.
- Ajoutez ou réduisez le nombre de partitions dans un flux à l'aide UpdateShardCountde.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon Kinesis Data Streams, consultez le manuel du développeur Amazon Kinesis Data Streams.

Répertorier les fragments de flux de données

Répertorier les détails de jusqu'à 100 partitions dans un flux spécifique.

Pour répertorier les partitions d'un flux de données Kinesis, utilisez l'ListShardsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Ajouter d'autres fragments de flux de données

Si vous avez besoin de davantage de partitions de flux de données, vous pouvez augmenter le nombre actuel de partitions. Nous vous recommandons de doubler votre nombre de partition en cas d’augmentation. Ainsi, chaque partition actuellement disponible est copiée afin d'augmenter votre capacité. Vous ne pouvez doubler le nombre de vos partitions que deux fois par période de 24 heures.

N'oubliez pas que la facturation pour l'utilisation de Kinesis Data Streams est calculée par partition. Par conséquent, lorsque la demande diminue, nous vous recommandons de réduire le nombre de partitions de moitié. Lorsque vous supprimez des partitions, vous pouvez uniquement réduire la quantité de partitions à la moitié de votre nombre de partitions actuel.

Pour mettre à jour le nombre de partitions d'un flux de données Kinesis, utilisez UpdateShardCountcette opération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création de flux de diffusion à l'aide de l'API Firehose et de la version 3 AWS SDK pour PHP

Amazon Data Firehose vous permet d'envoyer des données en temps réel à d'autres AWS services, notamment Amazon Kinesis Data Streams, Amazon S3, Amazon OpenSearch Service (OpenSearch Service) et Amazon Redshift, ou à Splunk. Créer un producteur de données avec les flux de diffusion

pour transférer des données vers la destination configurée chaque fois que vous ajoutez des données.

Les exemples suivants montrent comment :

- Créez un flux de diffusion à l'aide de CreateDeliveryStream.
- Obtenez des informations sur un seul flux de diffusion à l'aide de DescribeDeliveryStream.
- Répertoriez vos flux de diffusion à l'aide de ListDeliveryStreams.
- Envoyez des données à un flux de diffusion à l'aide de PutRecord.
- Supprimez un flux de diffusion à l'aide de DeleteDeliveryStream.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon Data Firehose, consultez le manuel du développeur Amazon Kinesis Data Firehose.

Création d'un flux de diffusion à l'aide d'un flux de données Kinesis

Pour établir un flux de diffusion qui place les données dans un flux de données Kinesis existant, utilisez l'CreateDeliveryStreamopération.

Cela permet aux développeurs de migrer les services Kinesis existants vers Firehose.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création d'un flux de diffusion à l'aide d'un compartiment Amazon S3

Pour établir un flux de diffusion qui place les données dans un compartiment Amazon S3 existant, utilisez l'CreateDeliveryStreamopération.

Indiquez les paramètres de destination, comme indiqué dans la section Paramètres de destination. Assurez-vous ensuite d'accorder à Firehose l'accès à votre compartiment Amazon S3, comme décrit dans Accorder à Kinesis Data Firehose l'accès à une destination Amazon S3.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création d'un flux de diffusion à l'aide OpenSearch de Service

Pour établir un flux de diffusion Firehose qui place les données dans un cluster de OpenSearch services, utilisez l'CreateDeliveryStreamopération.

Indiquez les paramètres de destination, comme indiqué dans la section Paramètres de destination. Assurez-vous d'accorder à Firehose l'accès à votre cluster de OpenSearch services, comme décrit dans Accorder à Kinesis Data Firehose l'accès à une destination Amazon ES.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Récupérez un flux de diffusion

Pour obtenir des informations sur un flux de diffusion Firehose existant, utilisez l'DescribeDeliveryStreamopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier les flux de diffusion existants connectés à Kinesis Data Streams

Pour répertorier tous les flux de diffusion Firehose existants qui envoient des données à Kinesis Data Streams, utilisez l'opération. ListDeliveryStreams

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier les flux de diffusion existants envoyant des données à d'autres AWS services

Pour répertorier tous les flux de diffusion Firehose existants qui envoient des données à Amazon S3, OpenSearch Service, Amazon Redshift ou à Splunk, utilisez l'opération. ListDeliveryStreams

Importations

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

```
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Envoyer des données vers un flux de diffusion Firehose existant

Pour envoyer des données via un flux de diffusion Firehose vers la destination que vous avez spécifiée, utilisez l'PutRecordopération après avoir créé un flux de diffusion Firehose.

Avant d'envoyer des données à un flux de diffusion Firehose, utilisez-le `DescribeDeliveryStream` pour vérifier si le flux de diffusion est actif.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un flux de diffusion Firehose

Pour supprimer un flux de diffusion Firehose, utilisez l'DeleteDeliveryStreamsopération. Ceci supprime également toutes les données envoyées dans le flux de diffusion.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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 MediaConvert exemples utilisant la AWS SDK pour PHP version 3

AWS Elemental MediaConvert est un service de transcodage vidéo basé sur des fichiers doté de fonctionnalités adaptées à la diffusion. Vous pouvez l'utiliser pour créer des ressources destinées à la diffusion et à la diffusion video-on-demand (VOD) sur Internet. Pour plus d'informations, consultez le Guide de l'utilisateur AWS Elemental MediaConvert.

L'API PHP pour AWS Elemental MediaConvert est exposée via la classe *`AWS.MediaConvert`*client. Pour plus d'informations, consultez `Class: AWS.MediaConvert`la référence de l'API.

Créez et gérez des tâches de transcodage dans AWS Elemental MediaConvert

Dans cet exemple, vous utilisez la AWS SDK pour PHP version 3 pour appeler AWS Elemental MediaConvert et créer une tâche de transcodage. Avant de commencer, vous devez télécharger la vidéo d'entrée dans le compartiment Amazon S3 que vous avez configuré pour le stockage d'entrée. Pour obtenir la liste des codecs et conteneurs vidéo d'entrée pris en charge, consultez la section Codecs et conteneurs d'entrée pris en charge dans le guide de l'utilisateur.AWS Elemental MediaConvert

Les exemples suivants montrent comment :

- Créez des tâches de transcodage dans. AWS Elemental MediaConvertCreateJob.

- Annulez une tâche de transcodage depuis la AWS Elemental MediaConvert file d'attente. CancelJob
- Récupérez le JSON pour une tâche de transcodage terminée. GetJob
- Récupérez un tableau JSON contenant jusqu'à 20 des dernières tâches créées. ListJobs

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour accéder au MediaConvert client, créez un rôle IAM qui donne AWS Elemental MediaConvert accès à vos fichiers d'entrée et aux compartiments Amazon S3 dans lesquels vos fichiers de sortie sont stockés. Pour plus de détails, consultez la section Configurer les autorisations IAM dans le guide de l'AWS Elemental MediaConvert utilisateur.

Création d'un client

Configurez le AWS SDK pour PHP en créant un MediaConvert client, avec la région de votre code. Dans cet exemple, la région est définie sur us-west-2.

Importations

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

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

Exemple de code

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

Définition d'une tâche de transcodage simple

Créez le code JSON qui définit les paramètres de tâche de transcodage.

Ces paramètres sont détaillés. Vous pouvez utiliser la AWS Elemental MediaConvert console pour générer les paramètres de tâche JSON en choisissant vos paramètres de tâche dans la console, puis en choisissant Afficher le JSON de la tâche en bas de la section Job. Cet exemple illustre le code JSON pour une tâche simple.

Exemple de code

```
$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"
    ]
];
```

Créez une tâche

Après avoir créé les paramètres de tâche JSON, appelez la méthode createJob en appelant un objet `AWS.MediaConvert service object` et en transmettant les paramètres. L'ID de la tâche créée est renvoyé dans les données de réponse.

Exemple de code

```
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";
}
```

Récupérer une offre d'emploi

Avec le JobID renvoyé lorsque vous avez appelé createjob, vous pouvez obtenir des descriptions détaillées des tâches récentes au format JSON.

Exemple de code

```
$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";
}
```

Annuler une tâche

Avec le JobID renvoyé lorsque vous avez appelé createjob, vous pouvez annuler une tâche pendant qu'elle est encore dans la file d'attente. Vous ne pouvez pas annuler des tâches qui ont déjà commencé le transcodage.

Exemple de code

```
$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";
}
```

Liste des travaux de transcodage récents

Créez le code JSON des paramètres, y compris les valeurs pour spécifier si vous souhaitez trier la liste dans l'ordre ASCENDING (croissant) ou DESCENDING (décroissant), l'ARN de la file d'attente de tâches à vérifier et le statut des tâches à inclure. Cela renvoie jusqu'à 20 tâches. Pour récupérer les 20 tâches les plus récentes suivantes, utilisez la chaîne nextToken renvoyée avec le résultat.

Exemple de code

```
$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";
}
```

Exemples d'Amazon S3 utilisant la AWS SDK pour PHP version 3

Amazon Simple Storage Service (Amazon S3) est un service Web qui fournit un stockage dans le cloud hautement évolutif. Amazon S3 fournit un stockage d'objets facile à utiliser, avec une interface de service Web simple permettant de stocker et de récupérer n'importe quel volume de données, où que vous soyez sur le Web.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Rubriques

- Création et utilisation de compartiments Amazon S3 avec la AWS SDK pour PHP version 3
- Gestion des autorisations d'accès au compartiment Amazon S3 avec la AWS SDK pour PHP version 3

- Configuration des compartiments Amazon S3 avec la AWS SDK pour PHP version 3
- Utilisation des téléchargements partitionnés sur Amazon S3 avec AWS SDK pour PHP la version 3
- URL pré-signée Amazon S3 avec AWS SDK pour PHP version 3
- Amazon S3 pré-signé POSTs avec la AWS SDK pour PHP version 3
- Utilisation d'un compartiment Amazon S3 en tant qu'hôte Web statique avec AWS SDK pour PHP la version 3
- Utilisation des politiques relatives aux compartiments Amazon S3 avec la AWS SDK pour PHP version 3
- Utilisation du point ARNs d'accès S3 (AWS SDK pour PHP version 3)
- Utilisez les points Multi-Region d'accès Amazon S3 avec AWS SDK pour PHP Version 3

Création et utilisation de compartiments Amazon S3 avec la AWS SDK pour PHP version 3

Les exemples suivants montrent comment :

- Renvoie une liste des buckets appartenant à l'expéditeur authentifié de la demande en utilisant. ListBuckets
- Créez un nouveau compartiment à l'aide de CreateBucket.
- Ajoutez un objet à un compartiment à l'aide de PutObject.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Importations

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

use Aws\S3\S3Client;
```

Lister les compartiments

Créez un fichier PHP avec le code suivant. Créez d'abord un service client AWS.S3 qui spécifie la AWS région et la version. Appelez ensuite la `listBuckets` méthode, qui renvoie tous les compartiments Amazon S3 appartenant à l'expéditeur de la demande sous la forme d'un tableau de structures de compartiments.

Exemple de code

```
$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";
}
```

Création d’un compartiment

Créez un fichier PHP avec le code suivant. Créez d'abord un service client AWS.S3 qui spécifie la AWS région et la version. Appelez ensuite la méthode `createBucket` avec un tableau comme paramètre. Le seul champ obligatoire est le compartiment (« Bucket ») clé, avec une valeur de chaîne pour le nom du compartiment à créer. Cependant, vous pouvez spécifier la AWS région à l'aide du champ CreateBucketConfiguration « ». Si elle aboutit, cette méthode renvoie l'emplacement (« Location ») du compartiment.

Exemple de code

```
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, 'amzn-s3-demo-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();
```

Placer un objet dans un seau

Pour ajouter des fichiers à votre nouveau compartiment, créez un fichier PHP avec le code suivant.

Dans votre ligne de commande, exécutez ce fichier et transmettez le nom du compartiment dans lequel vous souhaitez charger votre fichier sous forme de chaîne, suivi par le chemin d'accès complet du fichier à charger.

Exemple de code

```
$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";
}
```

Gestion des autorisations d'accès au compartiment Amazon S3 avec la AWS SDK pour PHP version 3

Les listes de contrôle d'accès (ACLs) sont l'une des options de politique d'accès basées sur les ressources que vous pouvez utiliser pour gérer l'accès à vos compartiments et objets. Vous pouvez l'utiliser ACLs pour accorder read/write des autorisations de base à d'autres AWS comptes. Pour en savoir plus, consultez la section Gestion de l'accès avec ACLs.

L'exemple suivant indique comment :

- Obtenez la politique de contrôle d'accès pour un bucket à l'aide de GetBucketAcl.
- Définissez les autorisations sur un bucket en utilisant ACLs, en utilisant PutBucketAcl.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Obtenir et définir une politique de liste de contrôle d'accès

Importations

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

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

Exemple de code

```

// 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";
}
```

Configuration des compartiments Amazon S3 avec la AWS SDK pour PHP version 3

Le partage des ressources cross-origin (CORS) définit un moyen pour les applications Web clientes chargées dans un domaine particulier d'interagir avec les ressources d'un autre domaine. Grâce à la prise en charge du CORS dans Amazon S3, vous pouvez créer des applications Web riches côté client avec Amazon S3 et autoriser de manière sélective un accès inter-origines à vos ressources Amazon S3.

Pour plus d'informations sur l'utilisation de la configuration CORS avec un compartiment Amazon S3, consultez Cross-Origin Resource Sharing (CORS).

Les exemples suivants montrent comment :

- Obtenez la configuration CORS d'un bucket à l'aide GetBucketCorsde.
- Définissez la configuration CORS pour un bucket à l'aide PutBucketCorsde.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Obtenez la configuration CORS

Créez un fichier PHP avec le code suivant. Commencez par créer un service client AWS.S3, puis appelez la méthode `getBucketCors` et spécifiez le compartiment dont vous souhaitez définir la configuration CORS.

Le seul paramètre obligatoire est le nom du compartiment sélectionné. Si le compartiment possède actuellement une configuration CORS, cette configuration est renvoyée par Amazon S3 sous forme d'CORSRules objet.

Importations

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

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

Exemple de code

```
$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());
}
```

Définissez la configuration CORS

Créez un fichier PHP avec le code suivant. Commencez par créer un service client AWS.S3. Appelez ensuite la `putBucketCors` méthode et spécifiez le bucket dont vous souhaitez définir la configuration CORS, puis CORSConfiguration sous forme d'objet CORSRules JSON.

Importations

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

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

Exemple de code

```
$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());
}
```

Utilisation des téléchargements partitionnés sur Amazon S3 avec AWS SDK pour PHP la version 3

Vous pouvez charger des objets de taille inférieure ou égale à 5 Go à l'aide d'une simple opération `PutObject`. Cependant, les méthodes de chargement partitionné (par exemple, `CreateMultipartUpload`, `UploadPart`, `CompleteMultipartUpload` et `AbortMultipartUpload`) vous permettent de charger des objets dont la taille est comprise entre 5 Mo et 5 To.

L'exemple suivant indique comment :

- Chargez un objet sur Amazon S3 à l'aide de ObjectUploader.
- Créez un téléchargement partitionné pour un objet Amazon S3 à l'aide MultipartUploaderde.
- Copiez des objets d'un emplacement Amazon S3 à un autre en utilisant ObjectCopier.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Uploader des objets

Si vous n'êtes pas sûr de la solution la mieux adaptée à la tâche, utilisez`ObjectUploader`. `PutObject` `MultipartUploader` `ObjectUploader`télécharge un fichier volumineux sur Amazon S3 en utilisant l'un ou l'autre `PutObject` ou `MultipartUploader` en fonction de la taille de la charge utile.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
```

```
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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);
```

Configuration

Le constructeur de l'objet `ObjectUploader` accepte les arguments suivants :

`$client`

L'objet `Aws\ClientInterface` à utiliser pour effectuer les transferts. Il doit s'agir d'une instance de `Aws\S3\S3Client`.

`$bucket`

(`string`, obligatoire) Nom du compartiment vers lequel l'objet est chargé.

`$key`

(`string`, obligatoire) Clé à utiliser pour l'objet concerné par le chargement.

`$body`

(`mixed`, obligatoire) Données d'objet à télécharger. Il peut s'agir `StreamInterface` d'une ressource de flux PHP ou d'une chaîne de données à télécharger.

`$acl`

(`string`) Liste de contrôle d'accès (ACL) à définir sur l'objet concerné par le chargement. Les objets sont privés par défaut.

`$options`

Un tableau associatif d'options de configuration pour le chargement partitionné. Les options de configuration suivantes sont valides :

`add_content_md5`

(`bool`) Réglez sur true pour calculer automatiquement la MD5 somme de contrôle pour le téléchargement.

`mup_threshold`

(`int`, par défaut :`int(16777216)`) Le nombre d'octets correspondant à la taille du fichier. Si la taille du fichier dépasse cette limite, un téléchargement partitionné est utilisé.

`before_complete`

(`callable`) Rappel à invoquer avant l'opération `CompleteMultipartUpload`. Le rappel doit avoir une signature de fonction similaire à :`function (Aws\Command $command) {...}`. Consultez la référence de l'CompleteMultipartUpload API pour connaître les paramètres que vous pouvez ajouter à l'`CommandInterface`objet.

`before_initiate`

(`callable`) Rappel à invoquer avant l'opération `CreateMultipartUpload`. Le rappel doit avoir une signature de fonction similaire à :`function (Aws\Command $command) {...}`. Le SDK invoque ce rappel si la taille du fichier dépasse la valeur. `mup_threshold` Consultez la référence de l'CreateMultipartUpload API pour connaître les paramètres que vous pouvez ajouter à l'`CommandInterface`objet.

`before_upload`

(`callable`) Rappel à invoquer avant `PutObject` toute `UploadPart` opération. Le rappel doit avoir une signature de fonction similaire à :`function (Aws\Command $command) {...}`. Le SDK invoque ce rappel si la taille du fichier est inférieure ou égale à la valeur. `mup_threshold` Consultez la référence de l'PutObject API pour connaître les paramètres que vous pouvez appliquer à la `PutObject` demande. Pour les paramètres qui s'appliquent à une `UploadPart` demande, consultez la référence de l'UploadPart API. Le SDK ignore tout paramètre non applicable à l'opération représentée par l'`CommandInterface`objet.

`concurrency`

(`int`, par défaut : `int(3)`) Nombre maximal d'opérations `UploadPart` simultanées autorisées pendant le chargement partitionné.

`part_size`

(`int`, par défaut : `int(5242880)`) Taille de partie, en octets, à utiliser lors d'un chargement partitionné. La valeur doit être comprise entre 5 Mo et 5 Go inclus.

`state`

(`Aws\Multipart\UploadState`) Objet représentant l'état du chargement partitionné et utilisé pour reprendre un chargement précédent. Lorsque cette option est fournie, les `$key` arguments `$bucket` et et l'`part_size`option sont ignorés.

`params`

Tableau associatif qui fournit des options de configuration pour chaque sous-commande. Par exemple :

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

MultipartUploader

Les chargements partitionnés sont conçus pour améliorer l'expérience de chargement pour les objets volumineux. Ils vous permettent de charger des parties d'objets indépendamment, dans n'importe quel ordre, et en parallèle.

Les clients Amazon S3 sont invités à utiliser les téléchargements partitionnés pour les objets de plus de 100 Mo.

MultipartUploader objet

Le kit SDK dispose d'un objet `MultipartUploader` spécial qui simplifie le processus de chargement partitionné.

Importations

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
$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";
}
```

Le chargeur crée un générateur de données partitionnées, en fonction de la source et de la configuration fournies, et tente de charger toutes les parties. Si le chargement de certaines parties échoue, le chargeur continue le chargement des autres parties jusqu'à ce que l'ensemble des données source soient lues. Par la suite, le chargeur tente de charger les parties ayant échoué ou lève une exception contenant des informations sur ces dernières.

Personnalisation d'un téléchargement en plusieurs parties

Vous pouvez définir des options personnalisées sur les opérations `CreateMultipartUpload`, `UploadPart` et `CompleteMultipartUpload` exécutées par le programme de chargement partitionné via des rappels transmis à son constructeur.

Importations

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

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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';
    },
]);
```

Collecte manuelle des déchets entre les chargements partiels

Si vous atteignez la limite de mémoire lors de chargements volumineux, c'est peut-être parce que des références cycliques générées par le kit SDK n'ont pas encore été traitées par le nettoyage de mémoire de PHP. L'appel manuel de l'algorithme de nettoyage entre les opérations peut permettre le traitement des cycles avant que cette limite ne soit atteinte. L'exemple suivant appelle l'algorithme de nettoyage en utilisant un rappel avant le chargement de chaque partie. Notez que l'appel du nettoyage de mémoire représente un coût en termes de performances, et que l'utilisation optimale dépendra de votre cas d'utilisation et de l'environnement.

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

Récupération après des erreurs

Lorsqu'une erreur se produit au cours du processus de chargement partitionné, une exception `MultipartUploadException` est levée. Cette exception donne accès à l'objet `UploadState`, qui

contient des informations sur la progression du chargement partitionné. L'objet `UploadState` peut être utilisé pour reprendre un chargement qui n'a pas pu aboutir.

Importations

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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);
}
```

Reprendre un chargement à partir d'un objet `UploadState` permet de tenter de charger les parties qui n'ont pas encore été chargées. L'objet d'état assure le suivi des parties manquantes, même si elles ne se suivent pas. Le chargeur lit ou recherche dans le fichier source fourni les plages d'octets appartenant aux parties qui n'ont pas encore été chargées.

Les objets `UploadState` étant sérialisables, vous pouvez également reprendre un chargement dans un processus différent. De plus, vous pouvez récupérer l'objet `UploadState`, même en l'absence d'exception, en appelant la méthode `$uploader->getState()`.

> **Important**
>
> Les flux transmis comme source à un `MultipartUploader` ne sont pas automatiquement rembobinés avant le chargement. Si vous utilisez un flux à la place d'un chemin d'accès dans une boucle similaire à celle de l'exemple précédent, réinitialisez la variable `$source` à l'intérieur du bloc `catch`.

Importations

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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);
```

Interruption d'un chargement partitionné

Un chargement partitionné peut être interrompu en récupérant le `UploadId` contenu dans l'objet `UploadState` et en le trnasmettant à `abortMultipartUpload`.

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

Chargements partitionnés asynchrones

Appeler `upload()` sur le `MultipartUploader` constitue une demande de blocage. Si vous travaillez dans un contexte asynchrone, vous pouvez obtenir une promesse pour le chargement partitionné.

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

use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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();
```

Configuration

Le constructeur de l'objet `MultipartUploader` accepte les arguments suivants :

`$client`

L'objet `Aws\ClientInterface` à utiliser pour effectuer les transferts. Il doit s'agir d'une instance de `Aws\S3\S3Client`.

`$source`

Les données source concernées par le chargement. Il peut s'agir d'un chemin d'accès ou d'une URL (par exemple, `/path/to/file.jpg`), d'un gestionnaire de ressources (par exemple, `fopen('/path/to/file.jpg', 'r')`) ou d'une instance d'un flux PSR-7.

`$config`

Un tableau associatif d'options de configuration pour le chargement partitionné.

Les options de configuration suivantes sont valides :

`acl`

(`string`) Liste de contrôle d'accès (ACL) à définir sur l'objet concerné par le chargement. Les objets sont privés par défaut.

before_complete

(callable) Rappel à invoquer avant l'opération CompleteMultipartUpload. Les rappels doivent avoir une signature de fonction telle que function (Aws\Command $command) {...}.

before_initiate

(callable) Rappel à invoquer avant l'opération CreateMultipartUpload. Les rappels doivent avoir une signature de fonction telle que function (Aws\Command $command) {...}.

before_upload

(callable) Rappel à invoquer avant toute opération UploadPart. Les rappels doivent avoir une signature de fonction telle que function (Aws\Command $command) {...}.

bucket

(string, obligatoire) Nom du compartiment vers lequel l'objet est chargé.

concurrency

(int, par défaut : int(5)) Nombre maximal d'opérations UploadPart simultanées autorisées pendant le chargement partitionné.

key

(string, obligatoire) Clé à utiliser pour l'objet concerné par le chargement.

part_size

(int, par défaut : int(5242880)) Taille de partie, en octets, à utiliser lors d'un chargement partitionné. Doit être comprise entre 5 Mo et 5 Go (inclus).

state

(Aws\Multipart\UploadState) Objet représentant l'état du chargement partitionné et utilisé pour reprendre un chargement précédent. Lorsque cette option est fournie, les options bucket, key et part_size sont ignorées.

add_content_md5

(boolean) Réglez sur true pour calculer automatiquement la MD5 somme de contrôle pour le téléchargement.

`params`

Tableau associatif qui fournit des options de configuration pour chaque sous-commande. Par exemple :

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

Copies en plusieurs parties

AWS SDK pour PHP Il inclut également un `MultipartCopy` objet qui est utilisé de la même manière que le`MultipartUploader`, mais qui est conçu pour copier des objets d'une taille comprise entre 5 Go et 5 To dans Amazon S3.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
use Aws\S3\S3Client;
```

Exemple de code

```
// 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";
```

```
}
```

URL pré-signée Amazon S3 avec AWS SDK pour PHP version 3

Vous pouvez authentifier certains types de demandes en transférant les informations requises en tant que paramètres de la chaîne de requête au lieu d'utiliser l'en-tête HTTP Authorization. Cela est utile pour permettre à un navigateur tiers d'accéder directement à vos données privées Amazon S3 par un navigateur tiers, sans transmettre la demande par proxy. L'idée est de créer une demande « pré-signée » et de l'encoder sous forme d'URL utilisable par un autre utilisateur. De plus, vous pouvez limiter une demande pré-signée en spécifiant un délai d'expiration.

Création d'une URL pré-signée pour une requête HTTP GET

L'exemple de code suivant montre comment créer une URL pré-signée pour une requête HTTP GET à l'aide du SDK pour PHP.

```
<?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;
```

La référence d'API pour la `createPresignedRequest` méthode fournit plus de détails.

Quelqu'un d'autre peut utiliser la `$presignedUrl` valeur pour récupérer l'objet dans l'heure qui suit. Lorsque la requête HTTP GET est effectuée, par exemple à l'aide d'un navigateur, le service S3 voit que l'appel provient de l'utilisateur qui a créé l'URL signée au préalable.

Création d'une URL pré-signée pour une requête HTTP PUT

L'exemple de code suivant montre comment créer une URL pré-signée pour une requête HTTP PUT à l'aide du SDK pour PHP.

```
<?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();
```

Une autre personne peut désormais utiliser l'URL pré-signée dans une requête HTTP PUT pour télécharger un fichier :

```
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;
    }
}
```

Amazon S3 pré-signé POSTs avec la AWS SDK pour PHP version 3

Tout comme le pré-signé URLs, le pré-signé vous POSTs permet de donner un accès d'écriture à un utilisateur sans lui donner AWS d'informations d'identification. Les formulaires POST pré-signés peuvent être créés à l'aide d'une instance d'AWSs3 V4. PostObject

Les exemples suivants montrent comment :

- Obtenez des données pour un formulaire de téléchargement POST d'un objet S3 à l'aide de la version PostObjectV4.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

> **Note**
>
> `PostObjectV4`ne fonctionne pas avec les informations d'identification provenant de AWS IAM Identity Center.

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Création de la PostObject V4

Pour créer une instance `PostObjectV4`, vous devez fournir les informations suivantes :

- Instance de `Aws\S3\S3Client`
- Compartiment
- Tableau associatif des champs de saisie du formulaire
- ensemble de conditions de politique (voir Construction des politiques dans le guide de l'utilisateur d'Amazon Simple Storage Service)
- Délai d'expiration de la chaîne pour la stratégie (facultatif, une heure par défaut).

Importations

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

Exemple de code

```
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>
```

Utilisation d'un compartiment Amazon S3 en tant qu'hôte Web statique avec AWS SDK pour PHP la version 3

Vous pouvez héberger un site web statique sur Amazon S3. Pour en savoir plus, consultez Hébergement d'un site Web statique sur Amazon S3.

L’exemple suivant indique comment :

- Obtenez la configuration du site Web pour un bucket à l'aide de GetBucketWebsite.
- Définissez la configuration du site Web pour un bucket à l'aide de PutBucketWebsite.
- Supprimez la configuration du site Web d'un compartiment à l'aide de DeleteBucketWebsite.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification. Voir Informations d'identification pour la AWS SDK pour PHP version 3.

Obtenir, définir et supprimer la configuration du site Web pour un bucket

Importations

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

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

Exemple de code

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

// Retrieving the Bucket Website Configuration
$bucket = 'amzn-s3-demo-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";
}
```

Utilisation des politiques relatives aux compartiments Amazon S3 avec la AWS SDK pour PHP version 3

Vous pouvez utiliser une politique de compartiment pour accorder des autorisations à vos ressources Amazon S3. Pour en savoir plus, consultez Utilisation de stratégies de compartiment et de stratégies utilisateur.

L'exemple suivant indique comment :

- Renvoie la politique pour un compartiment spécifié à l'aide de GetBucketPolicy.
- Remplacez une politique sur un bucket à l'aide de PutBucketPolicy.
- Supprimez une politique d'un bucket à l'aide de DeleteBucketPolicy.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Obtenir, supprimer et remplacer une politique sur un bucket

Importations

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

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

Exemple de code

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

$bucket = 'amzn-s3-demo-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";
}
```

Utilisation du point ARNs d'accès S3 (AWS SDK pour PHP version 3)

S3 a introduit des points d'accès, une nouvelle façon d'interagir avec les compartiments S3. Les points d'accès peuvent avoir des stratégies et une configuration uniques appliquées à eux plutôt que directement au compartiment. Vous AWS SDK pour PHP permet d'utiliser le point d'accès ARNs dans le champ du bucket pour les opérations d'API au lieu de spécifier explicitement le nom du bucket. Vous trouverez plus de détails sur le fonctionnement et le ARNs fonctionnement des points d'accès S3 ici. Les exemples suivants montrent comment :

- GetObjectÀ utiliser avec l'ARN d'un point d'accès pour récupérer un objet depuis un bucket.
- PutObjectÀ utiliser avec l'ARN d'un point d'accès pour ajouter un objet à un bucket.
- Configurez le client S3 pour qu'il utilise la région ARN au lieu de la région client.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Importations

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

use Aws\S3\S3Client;
```

Obtenir un objet

Créez d'abord un service client AWS.S3 qui spécifie la AWS région et la version. Ensuite, appelez la méthode `getObject` avec votre clé et un ARN de point d'accès S3 dans le champ `Bucket`, qui récupère l'objet du compartiment associé à ce point d'accès.

Exemple de code

```
$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'
]);
```

Placer un objet dans un seau

Créez d'abord un service client AWS.S3 qui spécifie la AWS région et la version. Ensuite, appelez la méthode `putObject` avec la clé, le corps ou le fichier source souhaité(e), et un ARN de point

d'accès S3 dans le champ `Bucket`, ce qui placera l'objet dans le compartiment associé à ce point d'accès.

Exemple de code

```
$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'
]);
```

Configurer le client S3 pour qu'il utilise la région ARN au lieu de la région cliente

Lors de l'utilisation d'un ARN de point d'accès S3 dans une opération cliente S3, par défaut, le client s'assure que la région ARN correspond à la région cliente, en lançant une exception si ce n'est pas le cas. Ce comportement peut être modifié pour accepter la région ARN sur la région client en définissant l'option de configuration `use_arn_region` sur `true`. Par défaut, l'option est définie sur `false`.

Exemple de code

```
$s3 = new S3Client([
    'version'        => 'latest',
    'region'         => 'us-west-2',
    'use_arn_region' => true
]);
```

Le client vérifiera également une variable d'environnement et une option de fichier de configuration, dans l'ordre de priorité suivant :

1. L'option client `use_arn_region`, comme dans l'exemple ci-dessus.
2. La variable d'environnement `AWS_S3_USE_ARN_REGION`

```
export AWS_S3_USE_ARN_REGION=true
```

1. La variable de configuration `s3_use_arn_region` dans le fichier de configuration AWS partagé (par défaut dans`~/.aws/config`).

```
[default]
s3_use_arn_region = true
```

Utilisez les points Multi-Region d'accès Amazon S3 avec AWS SDK pour PHP Version 3

Les points d' Multi-Region accès Amazon Simple Storage Service (S3) fournissent un point de terminaison global pour acheminer le trafic des requêtes Amazon S3 entre les deux Régions AWS.

Vous pouvez créer des points Multi-Region d'accès à l'aide du SDK pour PHP, d'un autre AWS SDK, de la console S3 ou AWS de la CLI,

> **Important**
>
> Pour utiliser les points Multi-Region d'accès avec le SDK pour PHP, l'extension AWS Common Runtime (AWS CRT) doit être installée dans votre environnement PHP.

Lorsque vous créez un point d' Multi-Region accès, Amazon S3 génère un Amazon Resource Name (ARN) au format suivant :

`arn:aws:s3::`*`account-id`*`:accesspoint/`*`MultiRegionAccessPoint_alias`*

Vous pouvez utiliser l'ARN généré à la place d'un nom de compartiment pour les `putObject()` méthodes `getObject()` et les méthodes.

```
<?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";
```

Gestion des secrets à l'aide de l'API Secrets Manager et de la AWS SDK pour PHP version 3

AWS Secrets Manager stocke et gère les secrets partagés tels que les mots de passe, les clés d'API et les informations d'identification de base de données. Avec le service Secrets Manager, les développeurs peuvent remplacer les informations d'identification codées en dur dans le code déployé par un appel intégré à Secrets Manager.

Secrets Manager prend en charge de manière native la rotation planifiée automatique des informations d'identification pour les bases de données Amazon Relational Database Service (Amazon RDS), renforçant ainsi la sécurité des applications. Secrets Manager peut également effectuer une rotation fluide des secrets pour d'autres bases de données et services tiers AWS Lambda afin d'implémenter des détails spécifiques aux services.

Les exemples suivants montrent comment :

- Créez un secret à l'aide de CreateSecret.
- Récupérez un secret en utilisant GetSecretValue.
- Répertoriez tous les secrets stockés par Secrets Manager à l'aide de ListSecrets.
- Obtenez des informations sur un secret spécifié à l'aide de DescribeSecret.
- Mettez à jour un secret spécifié à l'aide de PutSecretValue.
- Configurez une rotation secrète à l'aide de RotateSecret.
- Marquez un secret pour la suppression à l'aide de DeleteSecret.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Création d'un secret dans Secrets Manager

Pour créer un secret dans Secrets Manager, utilisez l'CreateSecretopération.

Dans cet exemple, un nom d'utilisateur et un mot de passe sont stockés sous forme de chaîne JSON.

Importations

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Récupérez un secret depuis Secrets Manager

Pour récupérer la valeur d'un secret stocké dans Secrets Manager, utilisez l'GetSecretValueopération.

Dans l'exemple suivant, `secret` il s'agit d'une chaîne contenant la valeur stockée. Si la valeur pour `username` est <<USERNAME>> et la valeur pour `password` est<<PASSWORD>>, le résultat de `secret` est :

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

`json_decode($secret, true)`Utilisez-le pour accéder aux valeurs du tableau.

Importations

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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;
```

Répertorier les secrets stockés dans Secrets Manager

Obtenez une liste de tous les secrets stockés par Secrets Manager à l'aide de cette ListSecretsopération.

Importations

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Récupérer les détails d'un secret

Les secrets stockés contiennent des métadonnées relatives aux règles de rotation, aux derniers accès ou aux dernières modifications dont ils on fait l'objet, aux balises créées par les utilisateurs, sans oublier l'Amazon Resource Name (ARN). Pour obtenir les détails d'un secret spécifié stocké dans Secrets Manager, utilisez l'DescribeSecretopération.

Importations

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Mettre à jour la valeur d'un secret

Pour stocker une nouvelle valeur secrète chiffrée dans Secrets Manager, utilisez l'PutSecretValueopération.

Cette opération crée une nouvelle version du secret. Si une version du secret existe déjà, ajoutez le paramètre `VersionStages` avec la valeur dans `AWSCURRENT` pour que la nouvelle valeur soit utilisée lors de la récupération de la valeur.

Importations

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Faites pivoter la valeur vers un secret existant dans Secrets Manager

Pour faire pivoter la valeur d'un secret existant stocké dans Secrets Manager, utilisez une fonction de rotation Lambda et l'RotateSecretopération.

Avant de commencer, créez une fonction Lambda pour faire pivoter votre secret. Le catalogue d'exemples de AWS code contient actuellement plusieurs exemples de code Lambda pour la rotation des informations d'identification des bases de données Amazon RDS.

> **Note**
>
> Pour plus d'informations sur la rotation des secrets, voir Rotation de vos AWS Secrets Manager secrets dans le guide de AWS Secrets Manager l'utilisateur.

Après avoir configuré votre fonction Lambda, configurez une nouvelle rotation secrète.

Importations

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Lorsqu'une rotation est configurée, vous pouvez implémenter une rotation à l'aide de cette RotateSecretopération.

Importations

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un secret dans Secrets Manager

Pour supprimer un secret spécifié de Secrets Manager, utilisez l'DeleteSecretopération. Pour éviter de supprimer accidentellement un secret, un DeletionDate tampon est automatiquement ajouté au secret pour indiquer le délai de restauration pendant lequel vous pouvez annuler la suppression. Si aucune durée n'est spécifiée pour la fenêtre de récupération, la durée par défaut est de 30 jours.

Importations

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

```
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Informations connexes

Les AWS SDK pour PHP exemples utilisent les opérations REST suivantes issues de la référence AWS Secrets Manager d'API :

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

Pour plus d'informations sur l'utilisation AWS Secrets Manager, consultez le guide de AWS Secrets Manager l'utilisateur.

Exemples d'Amazon SES utilisant la AWS SDK pour PHP version 3

Amazon Simple Email Service (Amazon SES) est une plateforme de messagerie qui vous permet d'envoyer et de recevoir des e-mails de manière simple et économique en utilisant vos propres adresses e-mail et domaines. Pour plus d'informations sur Amazon SES, consultez le manuel du développeur Amazon SES.

AWS propose deux versions du service Amazon SES et, par conséquent, le SDK pour PHP propose deux versions du client SesClient: et SESV2Client. Les fonctionnalités des clients se chevauchent dans de nombreux cas, bien que la façon dont les méthodes sont appelées ou les résultats puissent différer. Les deux offrent APIs également des fonctionnalités exclusives, de sorte que vous pouvez utiliser les deux clients pour accéder à toutes les fonctionnalités.

Les exemples de cette section utilisent tous l'original,`SesClient`.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Rubriques

- Vérification de l'identité des e-mails à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3
- Création de modèles d'e-mails personnalisés à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3
- Gestion des filtres d'e-mails à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3
- Création et gestion de règles de courrier électronique à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3
- Surveillance de votre activité d'envoi à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3
- Autorisation des expéditeurs à l'aide de l'API Amazon SES et de la version 3 AWS SDK pour PHP

Vérification de l'identité des e-mails à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3

Lorsque vous commencez à utiliser votre compte Amazon Simple Email Service (Amazon SES), tous les expéditeurs et destinataires doivent être vérifiés dans la AWS même région que celle à laquelle

vous envoyez des e-mails. Pour plus d'informations sur l'envoi d'e-mails, consultez la section Envoi d'e-mails avec Amazon SES.

Les exemples suivants montrent comment :

- Vérifiez une adresse e-mail à l'aide de VerifyEmailIdentity.
- Vérifiez un domaine de messagerie à l'aide de VerifyDomainIdentity.
- Répertoriez toutes les adresses e-mail en utilisant ListIdentities.
- Répertoriez tous les domaines de messagerie utilisant ListIdentities.
- Supprimez une adresse e-mail à l'aide de DeleteIdentity.
- Supprimez un domaine de messagerie à l'aide de DeleteIdentity.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Vérifier une adresse e-mail

Amazon SES peut envoyer des e-mails uniquement à partir d'adresses e-mail ou de domaines vérifiés. En vérifiant une adresse e-mail, vous démontrez que vous êtes le propriétaire de cette adresse et que vous souhaitez autoriser Amazon SES à envoyer des e-mails à partir de cette adresse.

Lorsque vous exécutez l'exemple de code suivant, Amazon SES envoie un e-mail à l'adresse que vous avez spécifiée. Lorsque vous (ou le destinataire de l'e-mail) cliquez sur le lien dans l'e-mail, l'adresse est vérifiée.

Pour ajouter une adresse e-mail à votre compte Amazon SES, utilisez l'VerifyEmailIdentityopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Vérifier un domaine de messagerie

Amazon SES peut envoyer des e-mails uniquement à partir d'adresses e-mail ou de domaines vérifiés. En vérifiant un domaine, vous démontrez que vous êtes le propriétaire de ce domaine. Lorsque vous validez un domaine, vous autorisez Amazon SES à envoyer des e-mails depuis n'importe quelle adresse de ce domaine.

Lorsque vous exécutez l'exemple de code suivant, Amazon SES vous fournit un jeton de vérification. Vous devez ajouter le jeton à la configuration DNS de votre domaine. Pour plus d'informations, consultez la section Vérifier un domaine auprès d'Amazon SES dans le manuel Amazon Simple Email Service Developer Guide.

Pour ajouter un domaine d'envoi à votre compte Amazon SES, utilisez l'VerifyDomainIdentityopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier les adresses e-mail

Pour récupérer une liste d'adresses e-mail soumises dans la AWS région actuelle, quel que soit le statut de vérification, utilisez l'ListIdentitiesopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Lister les domaines de messagerie

Pour récupérer une liste de domaines de messagerie soumis dans la AWS région actuelle, quel que soit le statut de vérification, utilisez l'ListIdentitiesopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer une adresse e-mail

Pour supprimer une adresse e-mail vérifiée de la liste des identités, utilisez l'DeleteIdentityopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un domaine de messagerie

Pour supprimer un domaine de messagerie vérifié de la liste des identités vérifiées, utilisez l'DeleteIdentityopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création de modèles d'e-mails personnalisés à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3

Amazon Simple Email Service (Amazon SES) vous permet d'envoyer des e-mails personnalisés pour chaque destinataire à l'aide de modèles. Les modèles incluent une ligne d'objet, ainsi que les parties texte et HTML du corps de l'e-mail. Les sections objet et corps peuvent également contenir des valeurs uniques personnalisées pour chaque destinataire.

Pour plus d'informations, consultez la section Envoi d'e-mails personnalisés à l'aide d'Amazon SES dans le manuel Amazon Simple Email Service Developer Guide.

Les exemples suivants montrent comment :

- Créez un modèle d'e-mail à l'aide de CreateTemplate.
- Répertoriez tous les modèles d'e-mails en utilisant ListTemplates.
- Récupérez un modèle d'e-mail à l'aide de GetTemplate.
- Mettez à jour un modèle d'e-mail à l'aide de UpdateTemplate.
- Supprimez un modèle d'e-mail à l'aide de DeleteTemplate.
- Envoyez un modèle d'e-mail en utilisant SendTemplatedEmail.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Créer un modèle d'e-mail

Pour créer un modèle pour envoyer des e-mails personnalisés, utilisez l'CreateTemplateopération. Le modèle peut être utilisé par n'importe quel compte autorisé à envoyer des messages dans la AWS région à laquelle le modèle est ajouté.

> **Note**
>
> Amazon SES ne valide pas votre code HTML. Assurez-vous donc qu'il HtmlPartest valide avant d'envoyer un e-mail.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Obtenez un modèle d'e-mail

Pour afficher le contenu d'un modèle d'e-mail existant, y compris la ligne d'objet, le corps HTML et le texte brut, utilisez l'GetTemplateopération. Seul TemplateName est requis.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier tous les modèles d'e-mails

Pour récupérer la liste de tous les modèles d'e-mail qui vous sont associés Compte AWS dans la AWS région actuelle, utilisez l'ListTemplatesopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Mettre à jour un modèle d'e-mail

Pour modifier le contenu d'un modèle d'e-mail spécifique, y compris la ligne d'objet, le corps HTML et le texte brut, utilisez l'UpdateTemplateopération.

Importations

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

```
use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un modèle d'e-mail

Pour supprimer un modèle d'e-mail spécifique, utilisez l'DeleteTemplateopération. Tout ce dont vous avez besoin, c'est du TemplateName.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Envoyer un e-mail avec un modèle

Pour utiliser un modèle pour envoyer un e-mail aux destinataires, utilisez l'SendTemplatedEmailopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Gestion des filtres d'e-mails à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3

Outre l'envoi d'e-mails, vous pouvez également recevoir des e-mails via Amazon Simple Email Service (Amazon SES). Un filtre d’adresses IP vous permet, le cas échéant, de choisir d’accepter ou de rejeter les messages provenant d’une adresse IP ou d’une plage d’adresses IP. Pour plus d’informations, consultez la section Gestion des filtres d’adresses IP pour la réception d’e-mails via Amazon SES.

Les exemples suivants montrent comment :

- Créez un filtre de courrier électronique à l'aide de CreateReceiptFilter.

- Répertoriez tous les filtres de courrier électronique utilisés ListReceiptFilters.
- Supprimez un filtre de courrier électronique à l'aide de DeleteReceiptFilter.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Création d'un filtre d'e-mail

Pour autoriser ou bloquer les e-mails provenant d'une adresse IP spécifique, utilisez l'CreateReceiptFilteropération. Indiquez l'adresse IP ou la plage d'adresses et un nom unique pour identifier ce filtre.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier tous les filtres d'e-mail

Pour répertorier les filtres d'adresse IP qui vous sont associés Compte AWS dans la AWS région actuelle, utilisez l'ListReceiptFiltersopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un filtre d'e-mail

Pour supprimer un filtre existant pour une adresse IP spécifique, utilisez l'DeleteReceiptFilteropération. Indiquez un nom de filtre unique pour identifier le filtre de réception à supprimer.

Si vous avez besoin de modifier la plage des adresses filtrées, vous pouvez supprimer un filtre de réception et en créer un nouveau.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création et gestion de règles de courrier électronique à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3

Outre l'envoi d'e-mails, vous pouvez également recevoir des e-mails via Amazon Simple Email Service (Amazon SES). Les règles de réception vous permettent de spécifier ce que fait Amazon SES avec les e-mails qu'il reçoit pour les adresses e-mail ou les domaines que vous possédez. Une règle peut envoyer des e-mails à d'autres AWS services, notamment Amazon S3, Amazon SNS ou. AWS Lambda

Pour plus d'informations, consultez les sections Gestion des ensembles de règles de réception pour la réception des e-mails Amazon SES et Gestion des règles de réception pour la réception des e-mails Amazon SES.

Les exemples suivants montrent comment :

- Créez un ensemble de règles de réception à l'aide de CreateReceiptRuleSet.
- Créez une règle de réception à l'aide de CreateReceiptRule.
- Décrivez un ensemble de règles de réception utilisant DescribeReceiptRuleSet.
- Décrivez une règle de réception à l'aide de DescribeReceiptRule.
- Répertoriez tous les ensembles de règles de réception en utilisant ListReceiptRuleSets.
- Mettez à jour une règle de réception à l'aide de UpdateReceiptRule.
- Supprimez une règle de réception à l'aide de DeleteReceiptRule.
- Supprimez un ensemble de règles de réception à l'aide de DeleteReceiptRuleSet.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Créer un jeu de règles de réception

Un ensemble de règles de réception renfermer plusieurs règles de réception. Au moins un ensemble de règles de réception doit être associé à votre compte avant de pouvoir créer une règle de réception. Pour créer un ensemble de règles de réception, indiquez un code unique RuleSetName et utilisez l'CreateReceiptRuleSetopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Création d'une règle de réception

Contrôlez vos e-mails entrants en ajoutant une règle de réception à un ensemble de règles de réception existant. Cet exemple vous montre comment créer une règle de réception qui envoie

des messages entrants à un compartiment Amazon S3, mais vous pouvez également envoyer des messages à Amazon SNS et. AWS Lambda Pour créer une règle de réception, fournissez une règle et le RuleSetName à l'CreateReceiptRuleopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Décrire un jeu de règles de réception

Une fois par seconde, renvoie les détails de l'ensemble de règles de réception spécifié. Pour utiliser l'DescribeReceiptRuleSetopération, fournissez le RuleSetName.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Décrire une règle de réception

Renvoie les détails d'une règle de réception spécifiée. Pour utiliser l'DescribeReceiptRuleopération, fournissez le RuleName et RuleSetName.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Répertorier tous les ensembles de règles de réception

Pour répertorier les ensembles de règles de réception qui existent sous votre nom Compte AWS dans la AWS région actuelle, utilisez l'ListReceiptRuleSetsopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Mettre à jour une règle de réception

Cet exemple montre comment mettre à jour une règle de réception qui envoie des messages entrants à une AWS Lambda fonction, mais vous pouvez également envoyer des messages à Amazon SNS et Amazon S3. Pour utiliser cette UpdateReceiptRuleopération, entrez la nouvelle règle de réception et le RuleSetName.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer un ensemble de règles de réception

Supprimez un ensemble de règles de réception qui n'est pas actuellement désactivé. Cette opération supprime également toutes les règles de réception qu'il contient. Pour supprimer un ensemble de règles de réception, fournissez le RuleSetName à l'DeleteReceiptRuleSetopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Supprimer une règle de réception

Pour supprimer une règle de réception spécifiée, fournissez le RuleName et RuleSetName à l'DeleteReceiptRuleopération.

Importations

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

use Aws\Exception\AwsException;
```

Exemple de code

```
$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";
}
```

Surveillance de votre activité d'envoi à l'aide de l'API Amazon SES et de la AWS SDK pour PHP version 3

Amazon Simple Email Service (Amazon SES) fournit des méthodes pour surveiller votre activité d'envoi. Nous vous recommandons d'implémenter ces méthodes afin d'assurer le suivi des métriques importantes, telles que les taux de retours à l'expéditeur, de réclamations et de rejets. Des taux de rebond et de plaintes trop élevés peuvent compromettre votre capacité à envoyer des e-mails via Amazon SES.

Les exemples suivants montrent comment :

- Vérifiez votre quota d'envoi à l'aide de GetSendQuota.
- Surveillez votre activité d'envoi à l'aide de GetSendStatistics.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Vérifiez votre quota d'envoi

Vous ne pouvez envoyer qu’une certaine quantité de messages sur une période de 24 heures. Pour vérifier le nombre de messages que vous êtes encore autorisé à envoyer, utilisez l'GetSendQuotaopération. Pour plus d’informations, consultez la section Gestion de vos limites d’envoi Amazon SES.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Surveillez votre activité d'envoi

Pour récupérer les statistiques des messages que vous avez envoyés au cours des deux dernières semaines, utilisez cette GetSendStatisticsopération. Cet exemple renvoie le nombre de tentatives d'envoi, de retours à l'expéditeur, de réclamations et de messages rejetés par tranche de 15 minutes.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Autorisation des expéditeurs à l'aide de l'API Amazon SES et de la version 3 AWS SDK pour PHP

Pour permettre à un autre Compte AWS, à un Gestion des identités et des accès AWS utilisateur ou à un AWS service d'envoyer des e-mails via Amazon Simple Email Service (Amazon SES) en votre nom, vous devez créer une politique d'autorisation d'envoi. Il s’agit d’un document JSON que vous attachez à une identité dont vous êtes propriétaire.

La stratégie mentionne expressément les entités que vous autorisez à effectuer des envois pour cette identité, ainsi que les conditions associées. Tous les expéditeurs, autres que vous et les entités auxquelles vous accordez explicitement des autorisations dans la stratégie, ne sont pas autorisés à envoyer des e-mails. Une identité peut avoir zéro, une ou plusieurs stratégies attachées. Une stratégie peut également contenir plusieurs instructions pour produire l'effet de plusieurs stratégies.

Pour plus d’informations, consultez la page Utilisation de l’autorisation d’envoi avec Amazon SES.

Les exemples suivants montrent comment :

- Créez un expéditeur autorisé à l'aide de PutIdentityPolicy.
- Récupérez les politiques d'un expéditeur autorisé à l'aide de GetIdentityPolicies.
- Répertoriez les expéditeurs autorisés à l'aide de ListIdentityPolicies.
- Révoquer l'autorisation accordée à un expéditeur autorisé à utiliser DeleteIdentityPolicy.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Pour plus d'informations sur l'utilisation d'Amazon SES, consultez le manuel du développeur Amazon SES.

Création d'un expéditeur autorisé

Pour autoriser une autre Compte AWS personne à envoyer des e-mails en votre nom, utilisez une politique d'identité pour ajouter ou mettre à jour l'autorisation d'envoyer des e-mails à

partir de vos adresses e-mail ou domaines vérifiés. Pour créer une politique d'identité, utilisez l'PutIdentityPolicyopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Récupérez les politiques d'un expéditeur autorisé

Renvoyez les stratégies d’autorisation d’envoi associées à une identité d’e-mail spécifique ou à une identité de domaine. Pour obtenir l'autorisation d'envoi pour une adresse e-mail ou un domaine donné, utilisez l'GetIdentityPolicyopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Répertorier les expéditeurs autorisés

Pour répertorier les politiques d'autorisation d'envoi associées à une identité e-mail ou à une identité de domaine spécifique dans la AWS région actuelle, utilisez l'ListIdentityPoliciesopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Révoquer l'autorisation d'un expéditeur autorisé

Supprimez l'autorisation d'envoi permettant Compte AWS à une autre personne d'envoyer des e-mails avec une identité e-mail ou une identité de domaine en supprimant la politique d'identité associée à l'DeleteIdentityPolicyopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Exemple de code

```
$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";
}
```

Exemples d'Amazon SNS utilisant la version 3 AWS SDK pour PHP

Amazon Simple Notification Service (Amazon SNS) est un service Web qui coordonne et gère la livraison ou l'envoi de messages aux points de terminaison ou aux clients abonnés.

Sur Amazon SNS, il existe deux types de clients : les éditeurs (également appelés producteurs) et les abonnés (également appelés consommateurs). Les éditeurs communiquent de façon asynchrone avec les abonnés en produisant et en envoyant un message à une rubrique, qui est un point d'accès logique et un canal de communication. Les abonnés (serveurs Web, adresses e-mail, files d'attente Amazon SQS, AWS Lambda fonctions) consomment ou reçoivent le message ou la notification via l'un des protocoles pris en charge (Amazon SQS, HTTP/HTTPS URLs e-mail, AWS SMS Lambda) lorsqu'ils sont abonnés au sujet.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Rubriques

- Gestion des sujets dans Amazon SNS avec la version 3 AWS SDK pour PHP
- Gestion des abonnements dans Amazon SNS avec AWS SDK pour PHP la version 3
- Envoyer des SMS sur Amazon SNS avec la version 3 AWS SDK pour PHP

Gestion des sujets dans Amazon SNS avec la version 3 AWS SDK pour PHP

Pour envoyer des notifications à Amazon Simple Queue Service (Amazon SQS) HTTP/HTTPS URLs, par e-mail AWS Lambda ou par e-mail AWS SMS, vous devez d'abord créer une rubrique qui gère la distribution des messages à tous les abonnés de cette rubrique.

En termes de conception du modèle d’observateur, une rubrique correspond à l’objet. Une fois qu’une rubrique est créée, vous ajoutez des abonnés qui reçoivent automatiquement une notification lorsqu’un message est publié dans cette rubrique.

Pour en savoir plus sur l'abonnement à des rubriques, consultez la section Gestion des abonnements dans Amazon SNS AWS SDK pour PHP avec la version 3.

Les exemples suivants montrent comment :

- Créez une rubrique pour publier des notifications d'utilisation CreateTopic.

- Renvoie une liste des sujets du demandeur à l'aide ListTopicsde.
- Supprimez un sujet et tous ses abonnements à l'aide de DeleteTopic.
- Renvoie toutes les propriétés d'une rubrique en utilisant GetTopicAttributes.
- Autoriser le propriétaire d'une rubrique à attribuer une nouvelle valeur à un attribut de la rubrique en utilisant SetTopicAttributes.

Pour plus d'informations sur l'utilisation d'Amazon SNS, consultez la rubrique Attributs d'Amazon SNS relatifs à l'état de livraison des messages.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Créer une rubrique

Pour créer un sujet, utilisez l'CreateTopicopération.

Chaque nom de rubrique que vous Compte AWS trouverez doit être unique.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Listez vos sujets

Pour répertorier jusqu'à 100 sujets existants dans la AWS région actuelle, utilisez l'ListTopicsopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Supprimer une rubrique

Pour supprimer un sujet existant et tous ses abonnements, utilisez l'DeleteTopicopération.

Les messages qui n'ont pas encore été livrés aux abonnés seront également supprimés.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Obtenir les attributs des rubriques

Pour récupérer les propriétés d'une seule rubrique existante, utilisez l'GetTopicAttributesopération.

Importations

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

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Définir les attributs de la rubrique

Pour mettre à jour les propriétés d'une seule rubrique existante, utilisez l'SetTopicAttributesopération.

Vous ne pouvez définir que les attributs `Policy`, `DisplayName` et `DeliveryPolicy`.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Gestion des abonnements dans Amazon SNS avec AWS SDK pour PHP la version 3

Utilisez les rubriques Amazon Simple Notification Service (Amazon SNS) pour envoyer des notifications à Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, adresses e-mail, (), ou. AWS Server Migration Service AWS SMS AWS Lambda

Les abonnements sont attachés à une rubrique qui gère l'envoi de messages aux abonnés. Pour en savoir plus sur la création de rubriques, consultez la section Gestion des rubriques dans Amazon SNS avec la AWS SDK pour PHP version 3.

Les exemples suivants montrent comment :

- S'abonner à une rubrique existante à l'aide de Subscribe.
- Vérifiez un abonnement à l'aide de ConfirmSubscription.
- Répertoriez les abonnements existants en utilisant ListSubscriptionsByTopic.
- Supprimer un abonnement à l'aide de Unsubscribe.
- Envoyer un message à tous les abonnés d'une rubrique à l'aide de Publish.

Pour plus d'informations sur l'utilisation d'Amazon SNS, consultez la section Utilisation d'Amazon System-to-System SNS pour la messagerie.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Abonner une adresse e-mail à une rubrique

Pour lancer un abonnement à une adresse e-mail, utilisez l'opération Subscribe.

Vous pouvez utiliser la méthode subscribe pour abonner plusieurs points de terminaison différents à une rubrique Amazon SNS, en fonction des valeurs utilisées pour les paramètres transmis. Cette action est présentée dans d'autres exemples de cette rubrique.

Dans cet exemple, le point de terminaison est une adresse e-mail. Un jeton de confirmation est envoyé à cette adresse e-mail. Vérifiez l'abonnement avec ce jeton de confirmation dans un délai de trois jours à compter de la réception.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Abonnement d'un point de terminaison d'application à une rubrique

Pour lancer un abonnement à une application Web, utilisez l'opération Subscribe.

Vous pouvez utiliser la méthode subscribe pour abonner plusieurs points de terminaison différents à une rubrique Amazon SNS, en fonction des valeurs utilisées pour les paramètres transmis. Cette action est présentée dans d'autres exemples de cette rubrique.

Dans cet exemple, le point de terminaison est une URL. Un jeton de confirmation est envoyé à cette adresse Web. Vérifiez l'abonnement avec ce jeton de confirmation dans un délai de trois jours à compter de la réception.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Abonner une fonction Lambda à une rubrique

Pour initier un abonnement à une fonction Lambda, utilisez l'opération Subscribe.

Vous pouvez utiliser la méthode subscribe pour abonner plusieurs points de terminaison différents à une rubrique Amazon SNS, en fonction des valeurs utilisées pour les paramètres transmis. Cette action est présentée dans d'autres exemples de cette rubrique.

Dans cet exemple, le point de terminaison est une fonction Lambda. Un jeton de confirmation est envoyé à cette fonction Lambda. Vérifiez l'abonnement avec ce jeton de confirmation dans un délai de trois jours à compter de la réception.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Abonnement d'un SMS à un sujet

Pour envoyer des messages SMS à plusieurs numéros de téléphone en même temps, abonnez chaque numéro à une rubrique.

Pour lancer un abonnement à un numéro de téléphone, utilisez l'opération Subscribe.

Vous pouvez utiliser la méthode subscribe pour abonner plusieurs points de terminaison différents à une rubrique Amazon SNS, en fonction des valeurs utilisées pour les paramètres transmis. Cette action est présentée dans d'autres exemples de cette rubrique.

Dans cet exemple, le point de terminaison est un numéro de téléphone au format E.164, une norme internationale de télécommunications.

Un jeton de confirmation est envoyé à ce numéro de téléphone. Vérifiez l'abonnement avec ce jeton de confirmation dans un délai de trois jours à compter de la réception.

Pour une autre méthode d'envoi de SMS avec Amazon SNS, consultez la section Envoi de messages SMS dans Amazon SNS avec AWS SDK pour PHP la version 3.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Confirmer l'abonnement à un sujet

Pour créer un abonnement, le propriétaire du point de terminaison doit confirmer l’intention de recevoir des messages envoyés par la rubrique à l’aide d’un jeton transmis lorsqu’un abonnement est établi initialement, comme indiqué précédemment. Les jetons de confirmation sont valides pendant trois jours. Au bout de trois jours, vous pouvez renvoyer un jeton en créant un nouvel abonnement.

Pour confirmer un abonnement, utilisez l'ConfirmSubscriptionopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Répertorier les abonnements à une rubrique

Pour répertorier jusqu'à 100 abonnements existants dans une AWS région donnée, utilisez l'ListSubscriptionsopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Annulation de l’abonnement à une rubrique

Pour supprimer un point de terminaison abonné à une rubrique, utilisez l’opération Unsubscribe.

Si l'abonnement nécessite une authentification pour être supprimé, seul le propriétaire de l'abonnement ou le propriétaire du sujet peut se désabonner, et une AWS signature est requise. Si l'appel de désabonnement ne nécessite pas l'authentification et si le demandeur n'est pas le propriétaire de l'abonnement, un message d'annulation final est remis au point de terminaison.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Publier un message sur une rubrique Amazon SNS

Pour envoyer un message à chaque point de terminaison abonné à une rubrique Amazon SNS, utilisez l'opération Publish.

Créez un objet contenant les paramètres de publication d'un message, notamment le texte du message et le nom de ressource Amazon (ARN) de la rubrique Amazon SNS.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Envoyer des SMS sur Amazon SNS avec la version 3 AWS SDK pour PHP

Vous pouvez utiliser Amazon Simple Notification Service (Amazon SNS) pour envoyer des SMS, ou des SMS, à des appareils compatibles SMS. Vous pouvez envoyer un message directement à un numéro de téléphone, ou vous pouvez envoyer un message à plusieurs numéros de téléphone simultanément en abonnant ces numéros de téléphone à une rubrique et en envoyant votre message à la rubrique.

Utilisez Amazon SNS pour définir vos préférences en matière de messagerie SMS, telles que la manière dont vos envois sont optimisés (en termes de coût ou de fiabilité), votre limite de dépenses mensuelles, la manière dont les envois de messages sont enregistrés et si vous souhaitez vous abonner aux rapports quotidiens d'utilisation des SMS. Ces préférences sont récupérées et définies sous forme d'attributs SMS pour Amazon SNS.

Lorsque vous envoyez un SMS, spécifiez le numéro de téléphone au format E.164. E.164 est une norme pour la structure des numéros de téléphone. Elle est utilisée pour les télécommunications internationales. Les numéros qui respectent ce format peuvent comporter 15 chiffres au maximum et commencent par le caractère plus (+) et le code pays. Par exemple, un numéro de téléphone américain au format E.164 s'affichera sous la forme +1001 0100XXX555.

Les exemples suivants montrent comment :

- Récupérez les paramètres par défaut pour l'envoi de SMS depuis votre compte à l'aide de Get SMSAttributes.
- Mettez à jour les paramètres par défaut pour l'envoi de SMS depuis votre compte à l'aide de Set SMSAttributes.
- Découvrez si le propriétaire d'un numéro de téléphone donné a choisi de ne pas recevoir de SMS de votre compte en utilisant CheckIfPhoneNumberISOptedOut.

- Répertoriez les numéros de téléphone pour lesquels le propriétaire a choisi de ne pas recevoir de SMS de votre compte en utilisant ListPhoneNumberOptedOut.
- Envoyer un message texte (SMS) directement à un numéro de téléphone à l'aide de Publish.

Pour plus d'informations sur l'utilisation d'Amazon SNS, consultez Utilisation d'Amazon SNS pour les notifications aux utilisateurs ayant un numéro de téléphone portable en tant qu'abonné (envoi de SMS).

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Obtenir les attributs des SMS

Pour récupérer les paramètres par défaut des messages SMS, utilisez l'SMSAttributesopération Obtenir.

Cet exemple permet d'obtenir l'attribut `DefaultSMSType`. Cet attribut contrôle si les messages SMS sont envoyés en tant que `Promotional`, ce qui optimise la transmission des messages au plus bas coût ou en tant que `Transactional`, ce qui optimise la transmission des messages à une fiabilité optimale.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Définir les attributs des SMS

Pour mettre à jour les paramètres par défaut des messages SMS, utilisez l'SMSAttributesopération Set.

Cet exemple définit l'attribut `DefaultSMSType` sur `Transactional`, ce qui optimise la transmission de message à une fiabilité optimale.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Vérifiez si un numéro de téléphone s'est désinscrit

Pour déterminer si le propriétaire d'un numéro de téléphone donné a choisi de ne pas recevoir de SMS depuis votre compte, utilisez cette CheckIfPhoneNumberIsOptedOutopération.

Dans cet exemple, le numéro de téléphone est au format E.164, une norme internationale de télécommunications.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Répertorier les numéros de téléphone désactivés

Pour récupérer la liste des numéros de téléphone pour lesquels le propriétaire a choisi de ne pas recevoir de SMS depuis votre compte, utilisez cette ListPhoneNumbersOptedOutopération.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Publier dans un message texte (message SMS)

Pour diffuser un message texte (SMS) directement à un numéro de téléphone, utilisez l'opération Publish.

Dans cet exemple, le numéro de téléphone est au format E.164, une norme internationale de télécommunications.

Les messages SMS peuvent contenir jusqu'à 140 octets. La limite de taille pour une action de publication de SMS est de 1 600 octets.

Pour plus d'informations sur l'envoi de messages SMS, consultez la section Envoi d'un message SMS.

Importations

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Exemple de code

```
$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());
}
```

Exemples d'Amazon SQS utilisant la version 3 AWS SDK pour PHP

Amazon Simple Queue Service (SQS) est un service de mise en file d'attente de messages rapide, fiable, évolutif et entièrement géré. Amazon SQS vous permet de dissocier les composants d'une application cloud. Amazon SQS inclut des files d'attente standard avec un débit et un at-least-once

traitement élevés, ainsi que des files d'attente FIFO qui fournissent une livraison FIFO (first>-in, first>out) et un traitement en une seule fois.

Tous les exemples de code pour la AWS SDK pour PHP version 3 sont disponibles ici GitHub.

Rubriques

- Activation des longues interrogations dans Amazon SQS avec AWS SDK pour PHP la version 3
- Gestion des délais de visibilité dans Amazon SQS AWS SDK pour PHP avec la version 3
- Envoyer et recevoir des messages dans Amazon SQS avec AWS SDK pour PHP la version 3
- Utilisation de files d'attente contenant des lettres mortes dans Amazon SQS avec la version 3 AWS SDK pour PHP
- Utilisation de files d'attente dans Amazon SQS AWS SDK pour PHP avec la version 3

Activation des longues interrogations dans Amazon SQS avec AWS SDK pour PHP la version 3

Les longues interrogations réduisent le nombre de réponses vides en permettant à Amazon SQS d'attendre pendant un certain temps qu'un message soit disponible dans la file d'attente avant d'envoyer une réponse. L'attente active de longue durée élimine également les fausses réponses vides en interrogeant tous les serveurs, à la place d'un simple échantillon. Pour activer l'attente active de longue durée, spécifiez un temps d'attente différent de zéro pour les messages reçus. Pour en savoir plus, consultez la section Attente active de longue durée SQS.

Les exemples suivants montrent comment :

- Définissez les attributs d'une file d'attente Amazon SQS pour permettre un long sondage, en utilisant. SetQueueAttributes
- Récupérez un ou plusieurs messages présentant un long sondage à l'aide de ReceiveMessage.
- Créez une longue file d'attente de sondage à l'aide de CreateQueue.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Définissez les attributs d'une file d'attente pour permettre un long sondage

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Récupérez les messages soumis à de longues interrogations

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Création d'une file d'attente avec un long sondage

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Gestion des délais de visibilité dans Amazon SQS AWS SDK pour PHP avec la version 3

Un délai de visibilité est une période pendant laquelle Amazon SQS empêche les autres composants consommateurs de recevoir et de traiter un message. Pour en savoir plus, consultez Délai de visibilité.

L'exemple suivant indique comment :

- Modifiez le délai de visibilité des messages spécifiés dans une file d'attente par de nouvelles valeurs, en utilisant ChangeMessageVisibilityBatch.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Modifier le délai de visibilité de plusieurs messages

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Envoyer et recevoir des messages dans Amazon SQS avec AWS SDK pour PHP la version 3

Pour en savoir plus sur les messages Amazon SQS, consultez les sections Envoyer un message à une file d'attente SQS et Recevoir et supprimer un message d'une file d'attente SQS dans le Guide de l'utilisateur de Service Quotas.

Les exemples suivants montrent comment :

- Envoyez un message à une file d'attente spécifiée à l'aide de SendMessage.
- Récupérez un ou plusieurs messages (jusqu'à 10) d'une file d'attente spécifiée à l'aide de ReceiveMessage.
- Supprimez un message d'une file d'attente à l'aide de DeleteMessage.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Envoyer un message

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Recevoir et supprimer des messages

Importations

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

use Aws\Exception\AwsException;
```

```
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Utilisation de files d'attente contenant des lettres mortes dans Amazon SQS avec la version 3 AWS SDK pour PHP

Une file d'attente de lettres mortes peut être ciblée par d'autres files d'attente (source) pour les messages ne pouvant pas être traités avec succès. Vous pouvez mettre de côté et isoler ces messages dans la file d’attente de lettres mortes pour déterminer pourquoi leur traitement a échoué.

Vous devez configurer individuellement chaque file d'attente source qui envoie des messages à une file d'attente de lettres mortes. Plusieurs files d'attente peuvent cibler une seule file d'attente de lettres mortes.

Pour en savoir plus, consultez la section Utilisation des files d'attente de lettres mortes SQS.

L'exemple suivant indique comment :

- Activez une file d'attente de lettres mortes à l'aide de. SetQueueAttributes

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d'identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called "Authentification avec AWS". Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called "Installation".

Activer une file d'attente de lettres mortes

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Utilisation de files d'attente dans Amazon SQS AWS SDK pour PHP avec la version 3

Pour en savoir plus sur les files d'attente Amazon SQS, consultez Comment fonctionnent les files d'attente SQS.

Les exemples suivants montrent comment :

- Renvoie une liste de vos files d'attente en utilisant ListQueues.
- Créez une nouvelle file d'attente en utilisant CreateQueue.
- Renvoie l'URL d'une file d'attente existante en utilisant GetQueueUrl.
- Supprimez une file d'attente spécifiée à l'aide de DeleteQueue.

Tous les exemples de code pour le AWS SDK pour PHP sont disponibles ici GitHub.

Informations d’identification

Avant d'exécuter l'exemple de code, configurez vos AWS informations d'identification, comme décrit dansthe section called “Authentification avec AWS”. Importez ensuite le AWS SDK pour PHP, comme décrit dansthe section called “Installation”.

Renvoie une liste de files d'attente

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Créer une file d'attente

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Renvoie l'URL d'une file d'attente

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Suppression d'une file d'attente

Importations

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Exemple de code

```
$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());
}
```

Envoyez des événements aux terminaux EventBridge mondiaux Amazon à l'aide du AWS SDK pour PHP Version 3

Vous pouvez utiliser les points de terminaison EventBridge mondiaux Amazon pour améliorer la disponibilité et la fiabilité de vos applications pilotées par les événements.

Une fois le point de terminaison EventBridge global configuré, vous pouvez lui envoyer des événements à l'aide du SDK pour PHP.

> **Important**
>
> Pour utiliser des points de terminaison EventBridge globaux avec le SDK pour PHP, l'extension AWS Common Runtime (AWS CRT) doit être installée dans votre environnement PHP.

L'exemple suivant utilise la PutEvents méthode `EventBridgeClient` pour envoyer un événement unique à un point de terminaison EventBridge global.

```
<?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]
]);
```

Ce billet de blog contient plus d'informations sur les points de terminaison EventBridge globaux.

Exemples de code SDK for PHP

Les exemples de code présentés dans cette rubrique vous montrent comment utiliser le AWS SDK pour PHP with AWS.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Certains services contiennent des exemples de catégories supplémentaires qui montrent comment tirer parti des bibliothèques ou des fonctions spécifiques au service.

Services

- Exemples API Gateway avec le kit SDK pour PHP
- Exemples Aurora avec le kit SDK pour PHP
- Exemples Auto Scaling avec le kit SDK pour PHP
- Exemples Amazon Bedrock avec le kit SDK pour PHP
- Exemples Amazon Bedrock Runtime avec le kit SDK pour PHP
- Exemples Amazon DocumentDB avec le kit SDK pour PHP
- Exemples DynamoDB avec le kit SDK pour PHP
- Exemples Amazon EC2 avec le kit SDK pour PHP
- AWS Glue exemples d'utilisation du SDK pour PHP
- Exemples IAM avec le kit SDK pour PHP
- Exemples Kinesis avec le kit SDK pour PHP
- AWS KMS exemples d'utilisation du SDK pour PHP
- Exemples Lambda avec le kit SDK pour PHP

- Exemples Amazon MSK avec le kit SDK pour PHP
- Exemples Amazon RDS avec le kit SDK pour PHP
- Exemples Amazon RDS Data Service avec le kit SDK pour PHP
- Exemples Amazon Rekognition avec le kit SDK pour PHP
- Exemples Amazon S3 avec le kit SDK pour PHP
- Exemples de compartiments de répertoires S3 avec le kit SDK pour PHP
- Exemples Amazon SES avec le kit SDK pour PHP
- Exemples Amazon SNS avec le kit SDK pour PHP
- Exemples Amazon SQS avec le kit SDK pour PHP

Exemples API Gateway avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK pour PHP with API Gateway.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Actions
- Scénarios

Actions

`GetBasePathMapping`

L'exemple de code suivant montre comment utiliser`GetBasePathMapping`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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();
```

- Pour plus de détails sur l'API, reportez-vous GetBasePathMappingà la section Référence des AWS SDK pour PHP API.

`ListBasePathMappings`

L'exemple de code suivant montre comment utiliser`ListBasePathMappings`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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();
```

- Pour plus de détails sur l'API, reportez-vous ListBasePathMappingsà la section Référence des AWS SDK pour PHP API.

`UpdateBasePathMapping`

L'exemple de code suivant montre comment utiliser`UpdateBasePathMapping`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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();
```

- Pour plus de détails sur l'API, reportez-vous UpdateBasePathMappingà la section Référence des AWS SDK pour PHP API.

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Exemples Aurora avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Aurora.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Scénarios

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit des éléments de travail dans une base de données Amazon Aurora sans serveur et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES).

Kit SDK pour PHP

Montre comment utiliser le AWS SDK pour PHP pour créer une application Web qui suit les éléments de travail dans une base de données Amazon RDS et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end intégré React.js pour interagir avec un backend PHP RESTful.

- Intégrez une application React.js Web à AWS des services.
- Répertoriez, ajoutez, mettez à jour et supprimez des éléments dans une table Amazon RDS.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples Auto Scaling avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK pour PHP with Auto Scaling.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Prise en main
- Principes de base
- Actions

Prise en main

Bonjour Auto Scaling

L'exemple de code suivant montre comment commencer à utiliser Auto Scaling.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- Pour plus de détails sur l'API, reportez-vous DescribeAutoScalingGroupsà la section Référence des AWS SDK pour PHP API.

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- créer un groupe Amazon EC2 Auto Scaling avec un modèle de lancement et des zones de disponibilité, et obtenir des informations sur l'exécution des instances ;
- Activez la collecte CloudWatch de métriques Amazon.
- mettre à jour la capacité souhaitée du groupe et attendre qu'une instance démarre ;
- résilier la plus ancienne instance du groupe ;
- répertorier les activités de mise à l'échelle qui se produisent en réponse aux demandes des utilisateurs et aux modifications de capacité ;
- Obtenez des statistiques pour CloudWatch les métriques, puis nettoyez les ressources.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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);
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Actions

`CreateAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`CreateAutoScalingGroup`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Pour plus de détails sur l'API, reportez-vous CreateAutoScalingGroupà la section Référence des AWS SDK pour PHP API.

`DeleteAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`DeleteAutoScalingGroup`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteAutoScalingGroupà la section Référence des AWS SDK pour PHP API.

`DescribeAutoScalingGroups`

L'exemple de code suivant montre comment utiliser`DescribeAutoScalingGroups`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DescribeAutoScalingGroupsà la section Référence des AWS SDK pour PHP API.

`DescribeAutoScalingInstances`

L'exemple de code suivant montre comment utiliser`DescribeAutoScalingInstances`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Pour plus de détails sur l'API, reportez-vous DescribeAutoScalingInstancesà la section Référence des AWS SDK pour PHP API.

`DescribeScalingActivities`

L'exemple de code suivant montre comment utiliser`DescribeScalingActivities`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Pour plus de détails sur l'API, reportez-vous DescribeScalingActivitiesà la section Référence des AWS SDK pour PHP API.

`DisableMetricsCollection`

L'exemple de code suivant montre comment utiliser`DisableMetricsCollection`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DisableMetricsCollectionà la section Référence des AWS SDK pour PHP API.

`EnableMetricsCollection`

L'exemple de code suivant montre comment utiliser`EnableMetricsCollection`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
```

```
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous EnableMetricsCollectionà la section Référence des AWS SDK pour PHP API.

`SetDesiredCapacity`

L'exemple de code suivant montre comment utiliser`SetDesiredCapacity`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous SetDesiredCapacityà la section Référence des AWS SDK pour PHP API.

`TerminateInstanceInAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`TerminateInstanceInAutoScalingGroup`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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;
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous TerminateInstanceInAutoScalingGroupà la section Référence des AWS SDK pour PHP API.

`UpdateAutoScalingGroup`

L'exemple de code suivant montre comment utiliser`UpdateAutoScalingGroup`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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);
}
```

- Pour plus de détails sur l'API, reportez-vous UpdateAutoScalingGroupà la section Référence des AWS SDK pour PHP API.

Exemples Amazon Bedrock avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon Bedrock.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Actions

Actions

`ListFoundationModels`

L'exemple de code suivant montre comment utiliser`ListFoundationModels`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Répertoriez les modèles de fondation Amazon Bedrock disponibles.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- Pour plus de détails sur l'API, reportez-vous ListFoundationModelsà la section Référence des AWS SDK pour PHP API.

Exemples Amazon Bedrock Runtime avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon Bedrock Runtime.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Amazon Nova
- Stable Image Core

Amazon Nova

Converse

L'exemple de code suivant montre comment envoyer un message texte à Amazon Nova à l'aide de l'API Converse de Bedrock.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Envoyez un message texte à Amazon Nova à l'aide de l'API Converse de Bedrock.

```
// 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();
```

- Pour plus de détails sur l'API, consultez Converse dans la Référence des API du kit AWS SDK pour PHP .

Stable Image Core

InvokeModel

L'exemple de code suivant montre comment invoquer Stability.ai Stable Image Core sur Amazon Bedrock pour générer une image.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez une image avec Stable Diffusion.

```
    public function invokeStableDiffusion(string $prompt, int $seed = 0, string
 $aspect_ratio = '1:1')
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available parameters 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-image-core-v1:1';
            $body = [
                'prompt' => $prompt,
                'aspect_ratio' => $aspect_ratio,
                'seed' => $seed,
                'output_format' => 'png',
            ];
```

```
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                '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;
    }
```

- Pour plus de détails sur l'API, reportez-vous InvokeModelà la section Référence des AWS SDK pour PHP API.

Exemples Amazon DocumentDB avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon DocumentDB.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Exemples sans serveur

Exemples sans serveur

Invocation d'une fonction Lambda à partir d'un déclencheur Amazon DocumentDB

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux de modifications DocumentDB. La fonction récupère les données utiles DocumentDB et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Amazon DocumentDB avec Lambda en utilisant PHP.

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
```

```
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Exemples DynamoDB avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide de DynamoDB.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Principes de base
- Actions
- Scénarios
- Exemples sans serveur

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- Créez une table pouvant contenir des données vidéo.

- Insérer, récupérez et mettez à jour un seul film dans la table.
- Écrivez des données vidéo dans la table à partir d'un exemple de fichier JSON.
- Recherchez les films sortis au cours d'une année donnée.
- Recherchez les films sortis au cours d'une plage d'années spécifique.
- Supprimez un film de la table, puis supprimez la table.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Étant donné que cet exemple utilise des fichiers de support, veillez à lire les instructions contenues dans le README.md fichier d'exemples PHP.

```
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);
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK pour PHP .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Interrogation
  - Analyser
  - UpdateItem

Actions

`BatchExecuteStatement`

L'exemple de code suivant montre comment utiliser`BatchExecuteStatement`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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,
                ],
            ],
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous BatchExecuteStatementà la section Référence des AWS SDK pour PHP API.

`BatchWriteItem`

L'exemple de code suivant montre comment utiliser`BatchWriteItem`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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;
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous BatchWriteItemà la section Référence des AWS SDK pour PHP API.

`CreateTable`

L'exemple de code suivant montre comment utiliser`CreateTable`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créer une table.

```
        $tableName = "ddb_demo_table_$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],
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateTableà la section Référence des AWS SDK pour PHP API.

`DeleteItem`

L'exemple de code suivant montre comment utiliser`DeleteItem`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteItemà la section Référence des AWS SDK pour PHP API.

`DeleteTable`

L'exemple de code suivant montre comment utiliser`DeleteTable`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTableà la section Référence des AWS SDK pour PHP API.

`ExecuteStatement`

L'exemple de code suivant montre comment utiliser`ExecuteStatement`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous ExecuteStatementà la section Référence des AWS SDK pour PHP API.

`GetItem`

L'exemple de code suivant montre comment utiliser`GetItem`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous GetItemà la section Référence des AWS SDK pour PHP API.

`ListTables`

L'exemple de code suivant montre comment utiliser`ListTables`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
```

```
            'ExclusiveStartTableName' => $exclusiveStartTableName,
            'Limit' => $limit,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListTablesà la section Référence des AWS SDK pour PHP API.

`PutItem`

L'exemple de code suivant montre comment utiliser`PutItem`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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);
    }
```

- Pour plus de détails sur l'API, reportez-vous PutItemà la section Référence des AWS SDK pour PHP API.

`Query`

L'exemple de code suivant montre comment utiliser`Query`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$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);
    }
```

- Pour plus de détails sur l'API, consultez Query dans la Référence des API du kit AWS SDK pour PHP .

`Analyser`

L'exemple de code suivant montre comment utiliser`Scan`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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);
    }
```

- Pour plus de détails sur l'API, consultez Scan dans la Référence des API du kit AWS SDK pour PHP .

`UpdateItem`

L'exemple de code suivant montre comment utiliser`UpdateItem`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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
                ]
            ],
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous UpdateItemà la section Référence des AWS SDK pour PHP API.

Scénarios

Création d’une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Interrogation d'une table à l'aide de lots d'instructions PartiQL

L'exemple de code suivant illustre comment :

- Obtenez un lot d'éléments en exécutant plusieurs instructions SELECT.
- Ajoutez un lot d'éléments en exécutant plusieurs instructions INSERT.
- Mettez à jour un lot d'éléments en exécutant plusieurs instructions UPDATE.
- Supprimez un lot d'éléments en exécutant plusieurs instructions DELETE.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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,
                ],
            ],
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous BatchExecuteStatementà la section Référence des AWS SDK pour PHP API.

Interrogation d’une table à l’aide de PartiQL

L’exemple de code suivant illustre comment :

- Obtenez un élément en exécutant une instruction SELECT.

- Ajoutez un élément en exécutant une instruction INSERT.
- Mettez à jour un élément en exécutant une instruction UPDATE.
- Supprimez un élément en exécutant une instruction DELETE.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous ExecuteStatementà la section Référence des AWS SDK pour PHP API.

Exemples sans serveur

Invocation d'une fonction Lambda à partir d'un déclencheur DynamoDB

L'exemple de code suivant montre comment mettre en œuvre une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux DynamoDB. La fonction récupère les données utiles DynamoDB et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement DynamoDB avec Lambda en utilisant PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
```

```
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur DynamoDB

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'un flux DynamoDB. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots DynamoDB avec Lambda à l'aide de PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
```

```
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

Exemples Amazon EC2 avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon EC2.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Actions

Actions

`CreateVpc`

L'exemple de code suivant montre comment utiliser`CreateVpc`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateVpcà la section Référence des AWS SDK pour PHP API.

`CreateVpcEndpoint`

L'exemple de code suivant montre comment utiliser`CreateVpcEndpoint`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateVpcEndpointà la section Référence des AWS SDK pour PHP API.

`DeleteVpc`

L'exemple de code suivant montre comment utiliser`DeleteVpc`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteVpcà la section Référence des AWS SDK pour PHP API.

`DeleteVpcEndpoints`

L'exemple de code suivant montre comment utiliser`DeleteVpcEndpoints`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteVpcEndpointsà la section Référence des AWS SDK pour PHP API.

`DescribeRouteTables`

L'exemple de code suivant montre comment utiliser`DescribeRouteTables`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /**
    * @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;
   }
```

- Pour plus de détails sur l'API, reportez-vous DescribeRouteTablesà la section Référence des AWS SDK pour PHP API.

AWS Glue exemples d'utilisation du SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour PHP with AWS Glue.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Principes de base
- Actions

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- Créez un robot d'exploration qui explore un compartiment Amazon S3 public et génère une base de données de CSV-formatted métadonnées.
- Répertoriez les informations relatives aux bases de données et aux tables de votre AWS Glue Data Catalog.
- Créez une tâche pour extraire les données CSV du compartiment S3, transformer les données et charger la JSON-formatted sortie dans un autre compartiment S3.
- Répertoriez les informations relatives aux exécutions de tâches, visualisez les données transformées et nettoyez les ressources.

Pour plus d'informations, consultez Tutoriel : prise en main de AWS Glue Studio.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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,
        ]);
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

Actions

`CreateCrawler`

L'exemple de code suivant montre comment utiliser`CreateCrawler`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
                        ]]
                ],
```

```
            ]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateCrawlerà la section Référence des AWS SDK pour PHP API.

`CreateJob`

L'exemple de code suivant montre comment utiliser`CreateJob`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
        ]);
```

```
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateJobà la section Référence des AWS SDK pour PHP API.

`DeleteCrawler`

L'exemple de code suivant montre comment utiliser`DeleteCrawler`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteCrawlerà la section Référence des AWS SDK pour PHP API.

`DeleteDatabase`

L'exemple de code suivant montre comment utiliser`DeleteDatabase`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteDatabaseà la section Référence des AWS SDK pour PHP API.

`DeleteJob`

L'exemple de code suivant montre comment utiliser`DeleteJob`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);
```

```
    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteJobà la section Référence des AWS SDK pour PHP API.

`DeleteTable`

L'exemple de code suivant montre comment utiliser`DeleteTable`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteTableà la section Référence des AWS SDK pour PHP API.

`GetCrawler`

L'exemple de code suivant montre comment utiliser`GetCrawler`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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,
            ]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous GetCrawlerà la section Référence des AWS SDK pour PHP API.

`GetDatabase`

L'exemple de code suivant montre comment utiliser`GetDatabase`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $databaseName = "doc-example-database-$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,
            ]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous GetDatabaseà la section Référence des AWS SDK pour PHP API.

`GetJobRun`

L'exemple de code suivant montre comment utiliser`GetJobRun`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous GetJobRunà la section Référence des AWS SDK pour PHP API.

`GetJobRuns`

L'exemple de code suivant montre comment utiliser`GetJobRuns`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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);
    }
```

- Pour plus de détails sur l'API, reportez-vous GetJobRunsà la section Référence des AWS SDK pour PHP API.

`GetTables`

L'exemple de code suivant montre comment utiliser`GetTables`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous GetTablesà la section Référence des AWS SDK pour PHP API.

`ListJobs`

L'exemple de code suivant montre comment utiliser`ListJobs`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListJobsà la section Référence des AWS SDK pour PHP API.

`StartCrawler`

L'exemple de code suivant montre comment utiliser`StartCrawler`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous StartCrawlerà la section Référence des AWS SDK pour PHP API.

`StartJobRun`

L'exemple de code suivant montre comment utiliser`StartJobRun`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        $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,
            ],
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous StartJobRunà la section Référence des AWS SDK pour PHP API.

Exemples IAM avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide de AWS SDK pour PHP with IAM.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Principes de base
- Actions

Principes de base

Principes de base

L'exemple de code suivant montre comment créer un utilisateur et endosser un rôle.

> **Warning**
>
> Afin d'éviter les risques de sécurité, n'employez pas les utilisateurs IAM pour l'authentification lorsque vous développez des logiciels spécialisés ou lorsque vous travaillez avec des données réelles. Préférez la fédération avec un fournisseur d'identité tel que AWS IAM Identity Center.

- Créer un utilisateur sans autorisation.
- Créer un rôle qui accorde l'autorisation de répertorier les compartiments Amazon S3 pour le compte.
- Ajouter une politique pour permettre à l'utilisateur d'assumer le rôle.
- Assumez le rôle et répertorier les compartiments S3 à l'aide d'informations d'identification temporaires, puis nettoyez les ressources.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Actions

`AttachRolePolicy`

L'exemple de code suivant montre comment utiliser`AttachRolePolicy`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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,
            ]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous AttachRolePolicyà la section Référence des AWS SDK pour PHP API.

`CreatePolicy`

L'exemple de code suivant montre comment utiliser`CreatePolicy`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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'];
    }
```

- Pour plus de détails sur l'API, reportez-vous CreatePolicyà la section Référence des AWS SDK pour PHP API.

`CreateRole`

L'exemple de code suivant montre comment utiliser`CreateRole`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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'];
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateRoleà la section Référence des AWS SDK pour PHP API.

`CreateServiceLinkedRole`

L'exemple de code suivant montre comment utiliser`CreateServiceLinkedRole`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateServiceLinkedRoleà la section Référence des AWS SDK pour PHP API.

`CreateUser`

L'exemple de code suivant montre comment utiliser`CreateUser`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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'];
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateUserà la section Référence des AWS SDK pour PHP API.

`GetAccountPasswordPolicy`

L'exemple de code suivant montre comment utiliser`GetAccountPasswordPolicy`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Pour plus de détails sur l'API, reportez-vous GetAccountPasswordPolicyà la section Référence des AWS SDK pour PHP API.

`GetPolicy`

L'exemple de code suivant montre comment utiliser`GetPolicy`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous GetPolicyà la section Référence des AWS SDK pour PHP API.

`GetRole`

L'exemple de code suivant montre comment utiliser`GetRole`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
```

```
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous GetRoleà la section Référence des AWS SDK pour PHP API.

`ListAttachedRolePolicies`

L'exemple de code suivant montre comment utiliser`ListAttachedRolePolicies`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListAttachedRolePoliciesà la section Référence des AWS SDK pour PHP API.

`ListGroups`

L'exemple de code suivant montre comment utiliser`ListGroups`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListGroupsà la section Référence des AWS SDK pour PHP API.

`ListPolicies`

L'exemple de code suivant montre comment utiliser`ListPolicies`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListPoliciesà la section Référence des AWS SDK pour PHP API.

`ListRolePolicies`

L'exemple de code suivant montre comment utiliser`ListRolePolicies`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous ListRolePoliciesà la section Référence des AWS SDK pour PHP API.

`ListRoles`

L'exemple de code suivant montre comment utiliser`ListRoles`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListRolesà la section Référence des AWS SDK pour PHP API.

`Répertorier les fournisseurs SAML`

L'exemple de code suivant montre comment utiliser`ListSAMLProviders`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Pour les détails de l'API, consultez ListSAMLProviders dans la Référence de l'API AWS SDK pour PHP .

`ListUsers`

L'exemple de code suivant montre comment utiliser`ListUsers`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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);
```

```
    }
```

- Pour plus de détails sur l'API, reportez-vous ListUsersà la section Référence des AWS SDK pour PHP API.

Exemples Kinesis avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide de Winesis.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Exemples sans serveur

Exemples sans serveur

Invoquer une fonction Lambda à partir d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux Kinesis. La fonction récupère la charge utile Kinesis, décode à partir de Base64 et enregistre le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Kinesis avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php
```

```
# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Kinesis

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'un flux Kinesis. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots Kinesis avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
```

```
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

AWS KMS exemples d'utilisation du SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l'aide du AWS SDK pour PHP with AWS KMS.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Prise en main
- Principes de base
- Actions

Prise en main

Bonjour AWS KMS

L'exemple de code suivant montre comment démarrer avec AWS Key Management Service.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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;
}
```

- Pour plus de détails sur l'API, reportez-vous ListKeysà la section Référence des AWS SDK pour PHP API.

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- créer une clé KMS ;
- répertorier les clés KMS de votre compte et obtenir des informations les concernant ;
- Activez et désactivez les clés KMS.
- générer une clé de données symétrique qui peut être utilisée pour le chiffrement côté client ;
- générer une clé asymétrique utilisée pour signer numériquement les données ;
- baliser des clés ;
- supprimer des clés KMS.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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;
        }
    }


}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CreateAlias
  - CreateGrant

- CreateKey
- Decrypt (Déchiffrer)
- DescribeKey
- DisableKey
- EnableKey
- Encrypt (Chiffrer)
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign (Signer)
- TagResource

Actions

`CreateAlias`

L'exemple de code suivant montre comment utiliser`CreateAlias`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
/***
 * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateAliasà la section Référence des AWS SDK pour PHP API.

`CreateGrant`

L'exemple de code suivant montre comment utiliser`CreateGrant`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateGrantà la section Référence des AWS SDK pour PHP API.

`CreateKey`

L'exemple de code suivant montre comment utiliser`CreateKey`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
```

```
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateKeyà la section Référence des AWS SDK pour PHP API.

`Decrypt`

L'exemple de code suivant montre comment utiliser`Decrypt`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, consultez Decrypt dans la Référence des API du kit AWS SDK pour PHP .

`DeleteAlias`

L'exemple de code suivant montre comment utiliser`DeleteAlias`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteAliasà la section Référence des AWS SDK pour PHP API.

`DescribeKey`

L'exemple de code suivant montre comment utiliser`DescribeKey`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DescribeKeyà la section Référence des AWS SDK pour PHP API.

`DisableKey`

L'exemple de code suivant montre comment utiliser`DisableKey`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DisableKeyà la section Référence des AWS SDK pour PHP API.

`EnableKey`

L'exemple de code suivant montre comment utiliser`EnableKey`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous EnableKeyà la section Référence des AWS SDK pour PHP API.

`Encrypt`

L'exemple de code suivant montre comment utiliser`Encrypt`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
```

```
    }
```

- Pour plus de détails sur l'API, consultez Encrypt dans la Référence des API du kit AWS SDK pour PHP .

`ListAliases`

L'exemple de code suivant montre comment utiliser`ListAliases`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ListAliasesà la section Référence des AWS SDK pour PHP API.

`ListGrants`

L'exemple de code suivant montre comment utiliser`ListGrants`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ListGrantsà la section Référence des AWS SDK pour PHP API.

`ListKeys`

L'exemple de code suivant montre comment utiliser`ListKeys`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ListKeysà la section Référence des AWS SDK pour PHP API.

`PutKeyPolicy`

L'exemple de code suivant montre comment utiliser`PutKeyPolicy`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous PutKeyPolicyà la section Référence des AWS SDK pour PHP API.

`RevokeGrant`

L'exemple de code suivant montre comment utiliser`RevokeGrant`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous RevokeGrantà la section Référence des AWS SDK pour PHP API.

`ScheduleKeyDeletion`

L'exemple de code suivant montre comment utiliser`ScheduleKeyDeletion`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous ScheduleKeyDeletionà la section Référence des AWS SDK pour PHP API.

`Sign (Signer)`

L'exemple de code suivant montre comment utiliser`Sign`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, consultez Sign dans la Référence des API du kit AWS SDK pour PHP .

`TagResource`

L'exemple de code suivant montre comment utiliser`TagResource`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /***
     * @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;
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous TagResourceà la section Référence des AWS SDK pour PHP API.

Exemples Lambda avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide de Lambda.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Principes de base
- Actions
- Scénarios
- Exemples sans serveur

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- Créer un rôle IAM et une fonction Lambda, puis charger le code du gestionnaire.
- Invoquer la fonction avec un seul paramètre et obtenir des résultats.
- Mettre à jour le code de la fonction et configurer avec une variable d'environnement.
- Invoquer la fonction avec de nouveaux paramètres et obtenir des résultats. Afficher le journal d'exécution renvoyé.
- Répertorier les fonctions pour votre compte, puis nettoyer les ressources.

Pour plus d'informations, consultez Créer une fonction Lambda à l'aide de la console.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
        $bucketName = "amzn-s3-demo-bucket-$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";
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Actions

`CreateFunction`

L'exemple de code suivant montre comment utiliser`CreateFunction`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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",
            ]);
        });
    }
```

- Pour plus de détails sur l'API, reportez-vous CreateFunctionà la section Référence des AWS SDK pour PHP API.

`DeleteFunction`

L'exemple de code suivant montre comment utiliser`DeleteFunction`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteFunctionà la section Référence des AWS SDK pour PHP API.

`GetFunction`

L'exemple de code suivant montre comment utiliser`GetFunction`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous GetFunctionà la section Référence des AWS SDK pour PHP API.

`Invocation`

L'exemple de code suivant montre comment utiliser`Invoke`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function invoke($functionName, $params, $logType = 'None')
    {
        return $this->lambdaClient->invoke([
            'FunctionName' => $functionName,
            'Payload' => json_encode($params),
            'LogType' => $logType,
        ]);
    }
```

- Pour en savoir plus sur l'API, consultez Invoke dans la Référence de l'API AWS SDK pour PHP .

`ListFunctions`

L'exemple de code suivant montre comment utiliser`ListFunctions`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    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,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous ListFunctionsà la section Référence des AWS SDK pour PHP API.

`UpdateFunctionCode`

L'exemple de code suivant montre comment utiliser`UpdateFunctionCode`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous UpdateFunctionCodeà la section Référence des AWS SDK pour PHP API.

`UpdateFunctionConfiguration`

L'exemple de code suivant montre comment utiliser`UpdateFunctionConfiguration`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- Pour plus de détails sur l'API, reportez-vous UpdateFunctionConfigurationà la section Référence des AWS SDK pour PHP API.

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Exemples sans serveur

Connexion à une base de données Amazon RDS dans une fonction Lambda

L'exemple de code suivant montre comment implémenter une fonction Lambda qui se connecte à une base de données RDS. La fonction effectue une simple requête de base de données et renvoie le résultat.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Connexion à une base de données Amazon RDS dans une fonction Lambda à l'aide de PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(
```

```
 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
           $connectionConfig['user'],
           $connectionConfig['password'],
           [
               PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
               PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
           ]
       );

       // Obtain the result of the query
       $stmt = $conn->prepare('SELECT ?+? AS sum');
       $stmt->execute([3, 2]);

       return $stmt->fetch(PDO::FETCH_ASSOC);
   }

   /**
    * @param mixed $event
    * @param Context $context
    * @return array
    */
   public function handle(mixed $event, Context $context): array
   {
       $this->logger->info("Processing query");

       // Execute database flow
       $result = $this->getQueryResults();

       return [
           'sum' => $result['sum']
       ];
   }
}

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

Invoquer une fonction Lambda à partir d'un déclencheur Kinesis

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux Kinesis. La

fonction récupère la charge utile Kinesis, décode à partir de Base64 et enregistre le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Kinesis avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
```

```
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

Invocation d'une fonction Lambda à partir d'un déclencheur DynamoDB

L'exemple de code suivant montre comment mettre en œuvre une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux DynamoDB. La fonction récupère les données utiles DynamoDB et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement DynamoDB avec Lambda en utilisant PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;
```

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
```

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

Invocation d'une fonction Lambda à partir d'un déclencheur Amazon DocumentDB

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un flux de modifications DocumentDB. La fonction récupère les données utiles DocumentDB et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Amazon DocumentDB avec Lambda en utilisant PHP.

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record
```

```
        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Invocation d'une fonction Lambda à partir d'un déclencheur Amazon MSK

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un cluster Amazon MSK. La fonction récupère les données utiles MSK et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Amazon MSK avec Lambda en utilisant PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\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);
```

Invoquer une fonction lambda à partir d'un déclencheur Amazon S3

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par le chargement d'un objet vers un compartiment S3. La fonction extrait le nom du compartiment S3 et la clé de l'objet à partir du paramètre d'événement et appelle l'API Amazon S3 pour récupérer et consigner le type de contenu de l'objet.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement S3 avec Lambda à l'aide de PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
```

```
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

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

Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages à partir d'une rubrique SNS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement SNS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Invoquer une fonction Lambda à partir d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages à partir d'une file d'attente SQS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement SQS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
```

```
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Kinesis

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'un flux Kinesis. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots Kinesis avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
```

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
```

```
    }
}

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur DynamoDB

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'un flux DynamoDB. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots DynamoDB avec Lambda à l'aide de PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
```

```
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'une file d'attente SQS. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots SQS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
```

```
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

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

Exemples Amazon MSK avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon MSK.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Exemples sans serveur

Exemples sans serveur

Invocation d'une fonction Lambda à partir d'un déclencheur Amazon MSK

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception d'enregistrements à partir d'un cluster Amazon MSK. La fonction récupère les données utiles MSK et journalise le contenu de l'enregistrement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement Amazon MSK avec Lambda en utilisant PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\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);
```

Exemples Amazon RDS avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon RDS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Actions
- Scénarios
- Exemples sans serveur

Actions

`CreateDBInstance`

L'exemple de code suivant montre comment utiliser`CreateDBInstance`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
}
```

- Pour plus d'informations sur l'API, consultez CreateDBInstance dans AWS SDK pour PHP API Reference.

`CreateDBSnapshot`

L'exemple de code suivant montre comment utiliser`CreateDBSnapshot`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
}
```

- Pour plus d'informations sur l'API, consultez CreateDBSnapshot dans la Référence d'API AWS SDK pour PHP .

`DeleteDBInstance`

L'exemple de code suivant montre comment utiliser`DeleteDBInstance`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
}
```

- Pour plus d'informations sur l'API, consultez DeleteDBInstance dans la Référence d'API AWS SDK pour PHP .

`DescribeDBInstances`

L'exemple de code suivant montre comment utiliser`DescribeDBInstances`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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";
}
```

- Pour plus d'informations sur l'API, consultez DescribeDBInstances dans la Référence d'API AWS SDK pour PHP .

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit des éléments de travail dans une base de données Amazon Aurora sans serveur et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES).

Kit SDK pour PHP

Montre comment utiliser le AWS SDK pour PHP pour créer une application Web qui suit les éléments de travail dans une base de données Amazon RDS et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end intégré React.js pour interagir avec un backend PHP RESTful.

- Intégrez une application React.js Web à AWS des services.
- Répertoriez, ajoutez, mettez à jour et supprimez des éléments dans une table Amazon RDS.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples sans serveur

Connexion à une base de données Amazon RDS dans une fonction Lambda

L'exemple de code suivant montre comment implémenter une fonction Lambda qui se connecte à une base de données RDS. La fonction effectue une simple requête de base de données et renvoie le résultat.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Connexion à une base de données Amazon RDS dans une fonction Lambda à l'aide de PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity
```

```
use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
```

```
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
```

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

Exemples Amazon RDS Data Service avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon RDS Data Service.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Scénarios

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit des éléments de travail dans une base de données Amazon Aurora sans serveur et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES).

Kit SDK pour PHP

Montre comment utiliser le AWS SDK pour PHP pour créer une application Web qui suit les éléments de travail dans une base de données Amazon RDS et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end intégré React.js pour interagir avec un backend PHP RESTful.

- Intégrez une application React.js Web à AWS des services.
- Répertoriez, ajoutez, mettez à jour et supprimez des éléments dans une table Amazon RDS.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples Amazon Rekognition avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon Rekognition.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Scénarios

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Exemples Amazon S3 avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon S3.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Prise en main
- Principes de base
- Actions
- Scénarios
- Exemples sans serveur

Prise en main

Hello Amazon S3

L'exemple de code suivant montre comment commencer à utiliser Amazon S3.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
use Aws\S3\S3Client;

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

- Pour plus de détails sur l'API, reportez-vous ListBucketsà la section Référence des AWS SDK pour PHP API.

Principes de base

Principes de base

L'exemple de code suivant illustre comment :

- créer un compartiment et y charger un fichier ;
- télécharger un objet à partir d'un compartiment ;
- copier un objet dans le sous-dossier d'un compartiment ;
- répertorier les objets d'un compartiment ;
- supprimer le compartiment et tous les objets qui y figurent.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        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";
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Actions

`CopyObject`

L'exemple de code suivant montre comment utiliser`CopyObject`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Copie simple d'un objet.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous CopyObjectà la section Référence des AWS SDK pour PHP API.

`CreateBucket`

L'exemple de code suivant montre comment utiliser`CreateBucket`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez un compartiment.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous CreateBucketà la section Référence des AWS SDK pour PHP API.

`DeleteBucket`

L'exemple de code suivant montre comment utiliser`DeleteBucket`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Supprimez un compartiment vide.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous DeleteBucketà la section Référence des AWS SDK pour PHP API.

`DeleteObject`

L'exemple de code suivant montre comment utiliser`DeleteObject`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   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;
       }
   }
```

- Pour plus de détails sur l'API, reportez-vous DeleteObjectà la section Référence des AWS SDK pour PHP API.

`DeleteObjects`

L'exemple de code suivant montre comment utiliser`DeleteObjects`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Supprimez un ensemble d'objets d'une liste de clés.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous DeleteObjectsà la section Référence des AWS SDK pour PHP API.

`GetObject`

L'exemple de code suivant montre comment utiliser`GetObject`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Obtenez un objet.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous GetObjectà la section Référence des AWS SDK pour PHP API.

`ListObjectsV2`

L'exemple de code suivant montre comment utiliser`ListObjectsV2`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Listez les objets dans un compartiment.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, voir ListObjectsV2 dans le manuel de référence des AWS SDK pour PHP API.

`PutObject`

L'exemple de code suivant montre comment utiliser`PutObject`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Chargez un objet dans un compartiment.

```
        $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.");
        }
```

- Pour plus de détails sur l'API, reportez-vous PutObjectà la section Référence des AWS SDK pour PHP API.

Scénarios

Créer une URL présignée

L'exemple de code suivant montre comment créer une URL présignée pour Amazon S3 et charger un objet.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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;
        }
    }

}
```

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Exemples sans serveur

Invoquer une fonction lambda à partir d'un déclencheur Amazon S3

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par le chargement d'un objet vers un compartiment S3. La fonction extrait le nom du compartiment S3 et la clé de l'objet à partir du paramètre d'événement et appelle l'API Amazon S3 pour récupérer et consigner le type de contenu de l'objet.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement S3 avec Lambda à l'aide de PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

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

Exemples de compartiments de répertoires S3 avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide des compartiments d'annuaire S3.

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d’un service.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l’exécution du code en contexte.

Rubriques

- Principes de base

Principes de base

Principes de base

L’exemple de code suivant illustre comment :

- Configurez un VPC et un point de terminaison d’un VPC.
- Configurez les politiques, les rôles et l’utilisateur pour qu’ils fonctionnent avec les compartiments de répertoires S3 et la classe de stockage S3 Express One Zone.
- Créez deux clients S3.
- Créez deux compartiments.
- créer un objet et le copier ;
- démontree la différence de performance ;
- remplir les compartiments pour montrer la différence lexicographique ;
- Invitez l’utilisateur à voir s’il souhaite nettoyer les ressources.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario illustrant les bases des compartiments de répertoires Amazon S3 et S3 Express One Zone.

```
        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;
        }
    }

}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObject
- GetObject
- ListObjects
- PutObject

Exemples Amazon SES avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon SES.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Scénarios

Scénarios

Créer un outil de suivi des éléments de travail sans serveur Aurora

L'exemple de code suivant montre comment créer une application Web qui suit des éléments de travail dans une base de données Amazon Aurora sans serveur et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES).

Kit SDK pour PHP

> Montre comment utiliser le AWS SDK pour PHP pour créer une application Web qui suit les éléments de travail dans une base de données Amazon RDS et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end intégré React.js pour interagir avec un backend PHP RESTful.

- Intégrez une application React.js Web à AWS des services.
- Répertoriez, ajoutez, mettez à jour et supprimez des éléments dans une table Amazon RDS.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Exemples Amazon SNS avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon SNS.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Actions
- Scénarios
- Exemples sans serveur

Actions

`CheckIfPhoneNumberIsOptedOut`

L'exemple de code suivant montre comment utiliser`CheckIfPhoneNumberIsOptedOut`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour plus de détails sur l'API, reportez-vous CheckIfPhoneNumberIsOptedOutà la section Référence des AWS SDK pour PHP API.

`ConfirmSubscription`

L'exemple de code suivant montre comment utiliser`ConfirmSubscription`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus de détails sur l'API, reportez-vous ConfirmSubscriptionà la section Référence des AWS SDK pour PHP API.

`CreateTopic`

L'exemple de code suivant montre comment utiliser`CreateTopic`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour plus de détails sur l'API, reportez-vous CreateTopicà la section Référence des AWS SDK pour PHP API.

`DeleteTopic`

L'exemple de code suivant montre comment utiliser`DeleteTopic`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteTopicà la section Référence des AWS SDK pour PHP API.

`GetSMSAttributes`

L'exemple de code suivant montre comment utiliser`GetSMSAttributes`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour de plus amples informations sur l'API, consultez GetSMSAttributes dans Référence de l'API AWS SDK pour PHP .

`GetTopicAttributes`

L'exemple de code suivant montre comment utiliser`GetTopicAttributes`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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());
}
```

- Pour plus de détails sur l'API, reportez-vous GetTopicAttributesà la section Référence des AWS SDK pour PHP API.

`ListPhoneNumbersOptedOut`

L'exemple de code suivant montre comment utiliser`ListPhoneNumbersOptedOut`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour plus de détails sur l'API, reportez-vous ListPhoneNumbersOptedOutà la section Référence des AWS SDK pour PHP API.

`ListSubscriptions`

L'exemple de code suivant montre comment utiliser`ListSubscriptions`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus de détails sur l'API, reportez-vous ListSubscriptionsà la section Référence des AWS SDK pour PHP API.

`ListTopics`

L'exemple de code suivant montre comment utiliser`ListTopics`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus de détails sur l'API, reportez-vous ListTopicsà la section Référence des AWS SDK pour PHP API.

`Publication`

L'exemple de code suivant montre comment utiliser`Publish`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour plus d'informations sur l'API, consultez Publier dans AWS SDK pour PHP Référence de l'API.

`SetSMSAttributes`

L'exemple de code suivant montre comment utiliser`SetSMSAttributes`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
$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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour les détails d'API, consultez SetSMSAttributes dans Référence de l'API AWS SDK pour PHP .

`SetTopicAttributes`

L'exemple de code suivant montre comment utiliser`SetTopicAttributes`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus de détails sur l'API, reportez-vous SetTopicAttributesà la section Référence des AWS SDK pour PHP API.

`S’abonner`

L'exemple de code suivant montre comment utiliser`Subscribe`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples de code AWS.

Abonnez une adresse e-mail à une rubrique.

```
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());
}
```

Abonnez un point de terminaison HTTP à une rubrique.

```
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());
}
```

- Pour plus de détails sur l'API, consultez Subscribe dans la Référence des API du kit AWS SDK pour PHP .

`Unsubscribe`

L'exemple de code suivant montre comment utiliser`Unsubscribe`.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour de plus amples informations sur l'API, consultez Se désabonner dans Référence de l'API AWS SDK pour PHP .

Scénarios

Création d'une application sans serveur pour gérer des photos

L'exemple de code suivant montre comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

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

Publier un message texte SMS

L'exemple de code suivant montre comment publier des messages texte à l'aide d'Amazon SNS.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
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());
}
```

- Pour plus d'informations, consultez le Guide du développeur AWS SDK pour PHP.
- Pour plus d'informations sur l'API, consultez Publier dans AWS SDK pour PHP Référence de l'API.

Exemples sans serveur

Invocation d'une fonction lambda à partir d'un déclencheur Amazon SNS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages à partir d'une rubrique SNS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Utilisation d'un événement SNS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
```

```
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Exemples Amazon SQS avec le kit SDK pour PHP

Les exemples de code suivants vous montrent comment effectuer des actions et implémenter des scénarios courants à l' AWS SDK pour PHP aide d'Amazon SQS.

Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code en contexte.

Rubriques

- Exemples sans serveur

Exemples sans serveur

Invoquer une fonction Lambda à partir d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment implémenter une fonction Lambda qui reçoit un événement déclenché par la réception de messages à partir d'une file d'attente SQS. La fonction extrait les messages du paramètre d'événement et consigne le contenu de chaque message.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Consommation d'un événement SQS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}
```

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

Signalement des échecs d'articles par lots pour les fonctions Lambda à l'aide d'un déclencheur Amazon SQS

L'exemple de code suivant montre comment mettre en œuvre une réponse partielle par lots pour les fonctions Lambda qui reçoivent des événements à partir d'une file d'attente SQS. La fonction signale les défaillances échecs d'articles par lots dans la réponse, en indiquant à Lambda de réessayer ces messages ultérieurement.

Kit SDK pour PHP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples sans serveur.

Signalement des échecs d'articles par lots SQS avec Lambda à l'aide de PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

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

Migration depuis la version 2 du AWS SDK pour PHP

Cette rubrique montre comment migrer votre code pour utiliser la version 3 du kit AWS SDK pour PHP et ce en quoi la nouvelle version diffère de la version 2 du kit SDK.

> **Note**
>
> Le modèle d'utilisation de base du kit SDK (par exemple, `$result = $client->operation($params);`) n'a pas changé entre la version 2 et la version 3, ce qui devrait aboutir à une migration transparente.

Introduction

La version 3 du AWS SDK pour PHP représente un effort important pour améliorer les fonctionnalités du SDK, intégrer plus de deux ans de commentaires clients, mettre à niveau nos dépendances, améliorer les performances et adopter les dernières normes PHP.

Quoi de neuf dans la version 3 ?

La version 3 AWS SDK pour PHP suit les normes PSR-4 et PSR-7 et suivra la norme à l'SemVeravenir.

Parmi les autres nouvelles fonctionnalités, citons :

- Système Middleware pour personnaliser le comportement du client de service
- Programmes de pagination flexibles pour l'itération sur les résultats paginés
- Possibilité d'interroger des données à partir d'objets de résultat et de paginateur avec JMESPath
- Débogage facile via l'option de configuration `'debug'`

Couche HTTP découplée

- Guzzle 6 est utilisé par défaut pour envoyer des demandes, mais Guzzle 5 est également pris en charge.
- Le kit SDK fonctionne dans les environnements où cURL n'est pas disponible.
- Les gestionnaires HTTP personnalisés sont également pris en charge.

Requêtes asynchrones

- Des fonctions telles que les programmes d'attente et le chargement partitionné peuvent également être utilisées de manière asynchrone.
- Il est possible de créer des workflows asynchrones à l'aide de promesses et coroutines.
- Les performances des demandes simultanées ou par lots sont améliorées.

Qu'est-ce qui est différent de la version 2 ?

Les dépendances du projet sont mises à jour

Les dépendances du kit SDK ont été modifiées dans cette version.

- Le SDK nécessite désormais PHP 8.1+. Nous utilisons abondamment les générateurs dans le code de kit SDK.
- Nous avons mis à niveau le SDK pour utiliser Guzzle 6 (ou 5), qui fournit l'implémentation du client HTTP sous-jacent utilisé par le SDK pour envoyer des requêtes aux services. AWS La dernière version de Guzzle amène un certain nombre d'améliorations, y compris les demandes asynchrones, les gestionnaires HTTP interchangeables, la conformité à PSR 7, de meilleures performances, et bien plus encore.
- Le package PSR-7 de PHP-FIG (`psr/http-message`) définit des interfaces pour représenter les requêtes HTTP, les réponses HTTP et les flux. URLs Ces interfaces sont utilisées dans le kit SDK et Guzzle, qui fournit l'interopérabilité avec d'autres packages conformes à PSR-7.
- L'implémentation par Guzzle de PSR-7 (`guzzlehttp/psr7`) fournit une implémentation des interfaces dans PSR-7, et plusieurs classes et fonctions utiles. Le kit SDK et Guzzle 6 s'appuient fortement sur ce package.
- L'implémentation des promesses/A+ de Guzzle (`guzzlehttp/promises`) est utilisée dans le kit SDK et Guzzle afin de fournir des interfaces pour la gestion des demandes et coroutines asynchrones. Alors que le gestionnaire HTTP multi-CURL de Guzzle implémente finalement le I/O modèle non bloquant qui permet les requêtes asynchrones, ce package permet de programmer selon ce paradigme. Voir Promises dans la AWS SDK pour PHP version 3 pour plus de détails.
- L'implémentation PHP de JMESPath(`mtdowling/jmespath.php`) est utilisée dans le SDK pour fournir la capacité d'interrogation de données des méthodes `Aws\Result::search()` and`Aws\ResultPaginator::search()`. Voir JMESPath Expressions dans la AWS SDK pour PHP version 3 pour plus de détails.

Les options de région et de version sont désormais obligatoires

Lors de l'instanciation d'un client pour n'importe quel service, spécifiez les options `'region'` et `'version'`. Dans la version 2 du AWS SDK pour PHP, `'version'` était totalement optionnel, et `'region'` était parfois facultatif. Dans la version 3, les deux sont toujours requis. Le fait d'indiquer clairement ces deux options vous permet de vous connecter à la version de l'API et à AWS la région dans lesquelles vous codez. Lorsque de nouvelles versions d'API sont créées ou que de nouvelles AWS régions sont disponibles, vous êtes isolé des modifications potentiellement importantes jusqu'à ce que vous soyez prêt à mettre à jour explicitement votre configuration.

> **Note**
>
> Si vous ne savez pas quelle version d'API vous utilisez, vous pouvez définir l'option `'version'` sur `'latest'`. Cependant, nous vous recommandons de définir les numéros de version d'API de façon explicite pour le code de production.
> Les services ne sont pas tous disponibles dans toutes les AWS régions. Pour connaître la liste des régions disponibles, consultez le document de référence Régions et points de terminaison.
> Pour les services disponibles uniquement via un seul point de terminaison global (par exemple, Amazon Route 53 et Amazon CloudFront) Gestion des identités et des accès AWS, instanciez les clients avec leur région configurée définie sur. `us-east-1`

> **Important**
>
> Le SDK inclut également des clients multirégionaux, qui peuvent envoyer des demandes à différentes AWS régions en fonction d'un paramètre (`@region`) fourni en tant que paramètre de commande. La région utilisée par défaut par ces clients est spécifiée avec l'option `region` fournie au constructeur client.

L'instanciation du client utilise le constructeur

Dans la version 3 de AWS SDK pour PHP, la façon dont vous instanciez un client a changé. A la place des méthodes `factory` de la version 2, vous pouvez simplement instancier un client à l'aide du mot clé `new`.

```
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**
>
> L'instanciation d'un client à l'aide de la méthode `factory()` fonctionne toujours. Cependant, elle est considérée comme obsolète.

La configuration du client a changé

Les options de configuration du client dans la version 3 du AWS SDK pour PHP ont légèrement changé par rapport à la version 2. Consultez la page Configuration pour la AWS SDK pour PHP version 3 pour une description de toutes les options prises en charge.

> **Important**
>
> Dans la version 3, les options `'key'` et `'secret'` ne sont plus valides au niveau de la racine, mais vous pouvez les transmettre dans le cadre de l'option `'credentials'`. L'une des raisons pour lesquelles nous l'avons fait était de décourager les développeurs de coder en dur leurs AWS informations d'identification dans leurs projets.

L'objet Sdk

La version 3 de AWS SDK pour PHP introduit l'`Aws\Sdk`objet en remplacement de`Aws\Common\Aws`. L'objet `Sdk` fonctionne comme une fabrique de clients. Il permet de gérer des options de configuration partagées sur plusieurs clients.

Bien que la classe `Aws` dans la version 2 du kit SDK fonctionnait comme un localisateur de service (il retournait toujours la même instance d'un client), la classe `Sdk` dans la version 3 renvoie une nouvelle instance d'un client à chaque fois qu'elle est utilisée.

De plus, l'objet `Sdk` ne prend pas en charge le même format de fichier de configuration à partir de la version 2 du kit SDK. Ce format de configuration a été spécifique à Guzzle 3 et est désormais obsolète. La configuration peut être effectuée plus simplement avec des tableaux de base. Elle est documentée dans la section Utilisation de la Classe Sdk.

Certains résultats de l'API ont changé

Pour garantir la cohérence de la manière dont le SDK analyse le résultat d'une opération d'API, Amazon, Amazon RDS et ElastiCache Amazon Redshift disposent désormais d'un élément d'encapsulation supplémentaire sur certaines réponses d'API.

Par exemple, l'appel du DescribeEngineDefaultParametersrésultat Amazon RDS dans la version 3 inclut désormais un élément d'encapsulage « EngineDefaults ». Dans la version 2, cet élément n'était pas présent.

```
$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'];
```

Les opérations suivantes sont impactées et contiennent désormais un élément d'encapsulage dans la sortie du résultat (fournis ci-dessous entre parenthèses) :

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
  - CopySnapshot (Instantané)

- CreateCacheCluster (CacheCluster)
- CreateCacheParameterGroup (CacheParameterGroup)
- CreateCacheSecurityGroup (CacheSecurityGroup)
- CreateCacheSubnetGroup (CacheSubnetGroup)
- CreateReplicationGroup (ReplicationGroup)
- CreateSnapshot (Instantané)
- DeleteCacheCluster (CacheCluster)
- DeleteReplicationGroup (ReplicationGroup)
- DeleteSnapshot (Instantané)
- DescribeEngineDefaultParameters (EngineDefaults)
- ModifyCacheCluster (CacheCluster)
- ModifyCacheSubnetGroup (CacheSubnetGroup)
- ModifyReplicationGroup (ReplicationGroup)
- PurchaseReservedCacheNodesOffering (ReservedCacheNode)
- RebootCacheCluster (CacheCluster)
- RevokeCacheSecurityGroupIngress (CacheSecurityGroup)

- Amazon RDS
  - AddSourceIdentifierToSubscription (EventSubscription)
  - Autoriser DBSecurity GroupIngress (DBSecuritygroupe)
  - Copier DBParameter un groupe (DBParameterGroupe)
  - Copier DBSnapshot (DBSnapshot)
  - CopyOptionGroup (OptionGroup)
  - Créer DBInstance (DBInstance)
  - Créer DBInstance ReadReplica (DBInstance)
  - Créer un DBParameter groupe (DBParameterGroupe)
  - Créer un DBSecurity groupe (DBSecurityGroupe)
  - Créer DBSnapshot (DBSnapshot)
  - Créer un DBSubnet groupe (DBSubnetGroupe)
  - CreateEventSubscription (EventSubscription)
  - CreateOptionGroup (OptionGroup)

- Supprimer DBInstance (DBInstance)
- Supprimer DBSnapshot (DBSnapshot)
- DeleteEventSubscription (EventSubscription)
- DescribeEngineDefaultParameters (EngineDefaults)
- Modifier DBInstance (DBInstance)
- Modifier DBSubnet le groupe (DBSubnetgroupe)
- ModifyEventSubscription (EventSubscription)
- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesOffre (réservéeDBInstance)
- Redémarrer DBInstance (DBInstance)
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- Restaurer DBInstance depuis DBSnapshot (DBInstance)
- Restaurer DBInstance ToPointInTime (DBInstance)
- Révoquer DBSecurity GroupIngress (DBSecuritygroupe)

- Amazon Redshift
  - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
  - AuthorizeSnapshotAccess (Instantané)
  - CopyClusterSnapshot (Instantané)
  - CreateCluster (Cluster)
  - CreateClusterParameterGroup (ClusterParameterGroup)
  - CreateClusterSecurityGroup (ClusterSecurityGroup)
  - CreateClusterSnapshot (Instantané)
  - CreateClusterSubnetGroup (ClusterSubnetGroup)
  - CreateEventSubscription (EventSubscription)
  - CreateHsmClientCertificate (HsmClientCertificate)
  - CreateHsmConfiguration (HsmConfiguration)
  - DeleteCluster (Cluster)
  - DeleteClusterSnapshot (Instantané)

  - DescribeDefaultClusterParameters (DefaultClusterParameters)

- DisableSnapshotCopy (Cluster)
- EnableSnapshotCopy (Cluster)
- ModifyCluster (Cluster)
- ModifyClusterSubnetGroup (ClusterSubnetGroup)
- ModifyEventSubscription (EventSubscription)
- ModifySnapshotCopyRetentionPeriod (Cluster)
- PurchaseReservedNodeOffering (ReservedNode)
- RebootCluster (Cluster)
- RestoreFromClusterSnapshot (Cluster)
- RevokeClusterSecurityGroupIngress (ClusterSecurityGroup)
- RevokeSnapshotAccess (Instantané)
- RotateEncryptionKey (Cluster)

Les classes Enum ont été supprimées

Nous avons supprimé les classes `Enum` (par exemple, `Aws\S3\Enum\CannedAcl`) qui existaient dans la version 2 du kit AWS SDK pour PHP. Enums étaient des classes concrètes au sein de l'API public du kit SDK qui contenaient des constantes représentant des groupes de valeurs de paramètre valides. Etant donné que les Enums sont spécifiques aux versions d'API, peuvent changer au fil du temps, créer des conflits avec les mots réservés PHP, et n'étaient finalement pas très utiles, nous les avons supprimées dans la version 3. Cela va dans le sens de la nature souple concernant l'exploitation des données et la version d'API de la version 3.

Au lieu d'utiliser des valeurs provenant d'objets `Enum`, utilisez directement les valeurs littérales (par exemple, `CannedAcl::PUBLIC_READ` → `'public-read'`).

Les classes d'exception détaillées ont été supprimées

Nous avons supprimé les classes Exception affinée qui existaient dans les espaces de noms de chaque service (par exemple, `Aws\Rds\Exception\{SpecificError}Exception`) pour des raisons similaires à celles pour lesquelles nous avons supprimé Enums. Les exceptions émises par un service ou une opération dépendent de la version d'API qui est utilisée (elles peuvent changer d'une version à l'autre). De plus, la liste complète des exceptions qui peuvent être émises par une opération donnée n'est pas disponible, ce qui rend les classes d'Exception affinées de la version 2 incomplètes.

Gérez les erreurs en interceptant la classe d'exception racine pour chaque service (par exemple, `Aws\Rds\Exception\RdsException`). Vous pouvez utiliser la méthode `getAwsErrorCode()` de l'exception pour rechercher des codes d'erreur spécifiques. Cette fonctionnalité est équivalente à celle de l'interception de différentes classes d'exception, mais fournit cette fonction sans ajouter la distension au kit SDK.

Les classes de façade statiques ont été supprimées

Dans la version 2 du AWS SDK pour PHP, il y avait une fonctionnalité obscure inspirée de Laravel qui vous permettait d'appeler `enableFacades()` la `Aws` classe pour activer un accès statique aux différents clients du service. Cette fonction est contraire aux bonnes pratiques PHP, et nous avons arrêté de la documenter il y a plus d'un an. Dans la version 3, cette fonction est complètement supprimée. Récupérez vos objets de client à partir de l'objet `Aws\Sdk` et utilisez-les en tant qu'instances de l'objet, et non en tant que classes statiques.

Les paginateurs remplacent les itérateurs

La version 2 de AWS SDK pour PHP avait une fonctionnalité nommée * iterators*. Il s'agissait d'objets qui étaient utilisés pour itérer sur des résultats paginés. On nous faisait souvent remarquer qu'ils n'étaient pas suffisamment flexibles, car l'itérateur émettait uniquement des valeurs spécifiques à partir de chaque résultat. Si vous aviez besoin d'autres valeurs à partir des résultats, il était possible de les récupérer uniquement via des écouteurs d'événements.

Dans la version 3, les itérateurs ont été remplacés par des programmes de pagination. Leurs objectifs sont similaires, mais les programmes de pagination sont plus flexibles. En effet, ils génèrent des objets de résultats au lieu de valeurs d'une réponse.

Les exemples suivants illustrent la façon dont les programmes de pagination sont différents des itérateurs, en démontrant comment récupérer des résultats paginés pour l'opération `S3 ListObjects` à la fois dans la version 2 et la version 3.

```
// 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";
    }
}
```

Les objets Paginator possèdent une `search()` méthode qui vous permet d'utiliser des JMESPathexpressions pour extraire plus facilement les données du jeu de résultats.

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

> **Note**
>
> La méthode `getIterator()` est toujours prise en charge pour permettre une transition en douceur vers la version 3, mais nous vous encourageons à migrer votre code pour permettre l'utilisation des programmes de pagination.

De nombreuses abstractions de haut niveau ont changé

En général, la plupart des abstractions de niveau supérieur (objets d'assistant spécifiques au service, hormis les clients) ont été améliorées ou mises à jour. Certaines ont été supprimées.

- Mis à jour:
  - La façon d'utiliser la fonction de chargement partitionné Amazon S3 a changé. Le téléchargement partitionné sur Amazon Glacier a été modifié de la même manière.
  - La façon de créer une version pré-signée d'Amazon S3 URLs a changé.
  - L'espace de noms `Aws\S3\Sync` a été remplacé par la classe `Aws\S3\Transfer`. Les méthodes `S3Client::uploadDirectory()` et `S3Client::downloadBucket()` sont toujours disponibles, mais présentent des options différentes. Consultez la documentation d'Amazon S3 Transfer Manager AWS SDK pour PHP version 3.
  - `Aws\S3\Model\ClearBucket` et `Aws\S3\Model\DeleteObjectsBatch` ont été remplacés par `Aws\S3\BatchDelete` et `S3Client::deleteMatchingObjects()`.
  - Les options et les comportements relatifs à l'utilisation du gestionnaire de session DynamoDB AWS SDK pour PHP avec la version 3 ont légèrement changé.

  - L'espace de noms `Aws\DynamoDb\Model\BatchRequest` a été remplacé par `Aws\DynamoDb\WriteRequestBatch`. Consultez la documentation de DynamoDB WriteRequestBatch.
  - Le `Aws\Ses\SesClient` gère désormais l'encodage base64 pour le `RawMessage` lors de l'utilisation de l'opération `SendRawEmail`.
- Supprimés :
  - Amazon `Item` DynamoDB`Attribute`,, `ItemIterator` et classes : elles étaient auparavant obsolètes dans la version 2.7.0.
  - Validateur de messages Amazon SNS : il s'agit désormais d'un projet distinct et léger qui ne nécessite pas le SDK comme dépendance. Ce projet est, toutefois, inclus dans les distributions Phar et ZIP du kit SDK. Vous trouverez un guide de démarrage sur le blog de développement AWS PHP.
  - Amazon S3 `AcpBuilder` et les objets associés ont été supprimés.

Comparaison d'échantillons de code provenant des deux versions du SDK

Les exemples suivants montrent en quoi l'utilisation de la version 3 de est AWS SDK pour PHP différente de la version 2.

Exemple : **`ListObjects`** opération Amazon S3

À partir de la version 2 du SDK

```
<?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";
}
```

À partir de la version 3 du SDK

Principales différences :

- Utilisation de `new` au lieu de `factory()` pour instancier le client.
- Les options `'version'` et `'region'` sont requises durant l'instanciation.

```
<?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";
}
```

Exemple : Instanciation d'un client avec une configuration globale

À partir de la version 2 du SDK

```
<?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.
```

À partir de la version 3 du SDK

Principales différences :

- Utilisation de la classe `Aws\Sdk` au lieu de la classe `Aws\Common\Aws`.
- Il n'y a pas de fichier de configuration. À la place, utilisation d'un tableau pour la configuration.
- L'option `'version'` est requise durant l'instanciation.
- Utilisation des méthodes `create<Service>()` au lieu de la méthode `get('<service>')`.

```
<?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.
```

Sécurité pour AWS SDK pour PHP

Chez Amazon Web Services (AWS), la sécurité dans le cloud est la priorité principale. En tant que client AWS , vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des organisations les plus pointilleuses sur la sécurité. La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud.

Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute tous les services proposés dans le AWS cloud et de vous fournir des services que vous pouvez utiliser en toute sécurité. Notre responsabilité en matière de sécurité est notre priorité absolue AWS, et l'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des programmes de AWS conformité.

Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez et par d'autres facteurs, notamment la sensibilité de vos données, les exigences de votre organisation et les lois et réglementations applicables.

Rubriques

- Protection des données dans AWS SDK pour PHP Version 3
- Gestion de l’identité et des accès
- Validation de conformité pour cela AWS Produit ou service
- De la résilience pour cela AWS Produit ou service
- Sécurité de l'infrastructure pour cela AWS Produit ou service
- Migration du client de chiffrement Amazon S3 (V1 vers V2) dans AWS SDK pour PHP Version 3
- Migration du client de chiffrement Amazon S3 (V2 vers V3) dans AWS SDK pour PHP Version 3

Protection des données dans AWS SDK pour PHP Version 3

Le modèle de responsabilité AWS partagée Le modèle de s'applique à la protection des données dans. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d’informations sur la

protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec la AWS SDK pour PHP version 3 ou une autre version Services AWS à l'aide de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Gestion de l’identité et des accès

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser AWS les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion de l’accès à l’aide de politiques
- Comment ? Services AWS collaborez avec IAM
- Résolution des problèmes AWS identité et accès

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction du travail que vous effectuez. AWS

Utilisateur du service : si vous avez l' Services AWS habitude de faire votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de nouvelles AWS fonctionnalités pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. Si vous comprenez bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS, consultez Résolution des problèmes AWS identité et accès le guide de l'utilisateur du Service AWS que vous utilisez.

Administrateur du service — Si vous êtes responsable des AWS ressources de votre entreprise, vous avez probablement un accès complet à AWS. C'est à vous de déterminer les AWS fonctionnalités et les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d’IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec AWS, consultez le guide de l'utilisateur Service AWS que vous utilisez.

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaiterez peut-être en savoir plus sur la façon d’écrire des politiques pour gérer l’accès à AWS. Pour consulter des exemples

de politiques AWS basées sur l'identité que vous pouvez utiliser dans IAM, consultez le guide de l'utilisateur Service AWS que vous utilisez.

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de votre annuaire d'entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle utilisateur à un rôle IAM (console) ou en appelant une opération AWS CLI ou AWS API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-based politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Resource-based politiques

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez Vue d'ensemble des listes de contrôle d'accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez Politiques de contrôle des ressources (RCP) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment ? Services AWS collaborez avec IAM

Pour obtenir une vue d'ensemble du Services AWS fonctionnement de la plupart des fonctionnalités IAM, consultez les AWS services compatibles avec IAM dans le guide de l'utilisateur IAM.

Pour savoir comment utiliser un service spécifique Service AWS avec IAM, consultez la section relative à la sécurité du guide de l'utilisateur du service concerné.

Résolution des problèmes AWS identité et accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans AWS
- Je ne suis pas autorisé à effectuer iam : PassRole

- Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mon AWS resources

Je ne suis pas autorisé à effectuer une action dans AWS

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `awes:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource *`my-example-widget`* à l'aide de l'action `awes:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans AWS. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mon AWS resources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises AWS en charge, consultezComment ? Services AWS collaborez avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Validation de conformité pour cela AWS Produit ou service

Notre nouvelle expérience AWS d'inscription n'est pas conçue pour les charges de travail réglementées. Si vous utilisez notre nouvelle expérience d' AWS inscription, mais que vous souhaitez l'utiliser AWS pour des charges de travail réglementées, vous pouvez vous inscrire à des fonctionnalités AWS (avancées) ou activer des fonctionnalités avancées pour votre AWS environnement.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Ce AWS produit ou service suit le modèle de responsabilité partagée par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité.

De la résilience pour cela AWS Produit ou service

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité.

Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant.

Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d’une zone à l’autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l’échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Ce AWS produit ou service suit le modèle de responsabilité partagée par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité.

Sécurité de l'infrastructure pour cela AWS Produit ou service

Ce AWS produit ou service utilise des services gérés et est donc protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à ce AWS produit ou service via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Ce AWS produit ou service suit le modèle de responsabilité partagée par le biais des services Amazon Web Services (AWS) spécifiques qu'il prend en charge. Pour obtenir des informations sur la sécurité des AWS services, consultez la AWS page de documentation sur la sécuritéAWS des services et les services concernés par les efforts de AWS conformité par programme de conformité.

Migration du client de chiffrement Amazon S3 (V1 vers V2) dans AWS SDK pour PHP Version 3

> **Note**
>
> Si vous utilisez la version 2 (V2) du client de chiffrement Amazon S3 et que vous souhaitez migrer vers la version 3 (V3), consultezMigration du client de chiffrement Amazon S3 (V2 vers V3) dans AWS SDK pour PHP Version 3.

Cette rubrique explique comment migrer vos applications de la version 1 (V1) du client de chiffrement Amazon Simple Storage Service (Amazon S3) vers la version 2 (V2), et comment garantir la disponibilité des applications tout au long du processus de migration.

Présentation de la migration

Cette migration s'effectue en deux phases :

1. Mettez à jour les clients existants pour lire les nouveaux formats. Déployez d'abord une version mise AWS SDK pour PHP à jour du dans votre application. Cela permet aux clients de chiffrement V1 existants de déchiffrer les objets écrits par les nouveaux clients V2. Si votre application utilise plusieurs AWS SDK, vous devez mettre à niveau chaque SDK séparément.

2. Migrez les clients de chiffrement et de déchiffrement vers la version V2. Une fois que tous vos clients de chiffrement V1 peuvent lire les nouveaux formats, vous pouvez migrer vos clients de chiffrement et de déchiffrement existants vers leurs versions V2 respectives.

Mettre à jour les clients existants pour lire les nouveaux formats

Le client de chiffrement V2 utilise des algorithmes de chiffrement que les anciennes versions du client ne prennent pas en charge. La première étape de la migration consiste à mettre à jour vos clients de déchiffrement V1 avec la dernière version du SDK. Une fois cette étape terminée, les clients V1 de votre application seront en mesure de déchiffrer les objets chiffrés par les clients de chiffrement V2. Voir les détails ci-dessous pour chaque version majeure du AWS SDK pour PHP.

Mise à niveau AWS SDK pour PHP Version 3

La version 3 est la dernière version du AWS SDK pour PHP. Pour terminer cette migration, vous devez utiliser la version 3.148.0 ou ultérieure du `aws/aws-sdk-php` package.

Installation depuis la ligne de commande

Pour les projets installés à l'aide de Composer, dans le fichier Composer, mettez à jour le package du SDK vers la version 3.148.0 du SDK, puis exécutez la commande suivante.

```
composer update aws/aws-sdk-php
```

Installation à l'aide du fichier Phar ou Zip

Utilisez l’une des méthodes suivantes. Veillez à placer le fichier SDK mis à jour à l'emplacement requis par votre code, qui est déterminé par l'instruction require.

Pour les projets installés à l'aide du fichier Phar, téléchargez le fichier mis à jour : `aws.phar`.

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

Pour les projets installés à l'aide du fichier Zip, téléchargez le fichier mis à jour : .

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

Migrer les clients de chiffrement et de déchiffrement vers la version V2

Après avoir mis vos clients à jour pour qu'ils lisent les nouveaux formats de chiffrement, vous pouvez mettre à jour vos applications avec les clients de chiffrement et de déchiffrement V2. Les étapes suivantes vous montrent comment migrer avec succès votre code de la V1 à la V2.

Exigences relatives à la mise à jour vers les clients V2

1. Le contexte de AWS KMS chiffrement doit être transmis aux `S3EncryptionClientV2::putObjectAsync` méthodes `S3EncryptionClientV2::putObject` et. AWS KMS un contexte de chiffrement est un tableau associatif de paires clé-valeur, que vous devez ajouter au contexte de chiffrement pour AWS KMS le chiffrement par clé. Si aucun contexte supplémentaire n'est requis, vous pouvez transmettre un tableau vide.

2. `@SecurityProfile`doit être transmis dans les `getObjectAsync` méthodes `getObject` et dans`S3EncryptionClientV2`. `@SecurityProfile`est un nouveau paramètre obligatoire des `getObject...` méthodes. Si ce paramètre est défini sur`'V2'`, seuls les objets dont le V2-compatible format est chiffré peuvent être déchiffrés. La définition de ce paramètre permet `'V2_AND_LEGACY'` également de déchiffrer les objets chiffrés V1-compatible au format. Pour prendre en charge la migration, définissez `@SecurityProfile` sur`'V2_AND_LEGACY'`. À utiliser `'V2'` uniquement pour le développement de nouvelles applications.

3. (facultatif) Incluez le `@KmsAllowDecryptWithAnyCmk` paramètre dans le `S3EncryptionClientV2::getObject` et `S3EncryptionClientV2::getObjectAsync*` `methods.` Un nouveau paramètre a été ajouté appelé`@KmsAllowDecryptWithAnyCmk`. La

définition de ce paramètre pour `true` activer le déchiffrement sans fournir de clé KMS. La valeur par défaut est `false`.

4. Pour le déchiffrement avec un client V2, si le `@KmsAllowDecryptWithAnyCmk` paramètre n'est pas défini sur `true` pour les appels de `"getObject..."` méthode, un `kms-key-id` doit être fourni au `KmsMaterialsProviderV2` constructeur.

Exemples de migration

Exemple 1 : migration vers des clients V2

Pre-migration

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Post-migration

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Exemple 2 : utilisation AWS KMS avec kms-key-id

> **Note**
>
> Ces exemples utilisent des importations et des variables définies dans l'exemple 1. Par exemple : `$encryptionClient`.

Pre-migration

```
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,
]);
```

Post-migration

```
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,
]);
```

Migration du client de chiffrement Amazon S3 (V2 vers V3) dans AWS SDK pour PHP Version 3

> **Note**
>
> Si vous utilisez la version 1 (V1) du client de chiffrement Amazon S3, vous devez d'abord migrer vers la version 2 (V2) avant de migrer vers la version 3 (V3). Consultez Migration du client de chiffrement Amazon S3 (V1 vers V2) dans AWS SDK pour PHP Version 3.

Cette rubrique explique comment migrer vos applications de la version 2 (V2) du client de chiffrement Amazon Simple Storage Service (Amazon S3) vers la version 3 (V3), et comment garantir la disponibilité des applications tout au long du processus de migration. La version 3 introduit AES GCM avec des politiques d'engagement et d'engagement clés pour améliorer la sécurité et protéger contre la falsification des clés de données.

Présentation de la migration

Cette migration s'effectue en deux phases :

1. Mettez à jour les clients existants pour lire les nouveaux formats. Déployez d'abord une version mise AWS SDK pour PHP à jour du dans votre application. Cela permet aux clients de chiffrement V2 existants de déchiffrer les objets écrits par les nouveaux clients V3. Si votre application utilise plusieurs AWS SDK, vous devez mettre à niveau chaque SDK séparément.

2. Migrez les clients de chiffrement et de déchiffrement vers la version 3. Une fois que tous vos clients de chiffrement V2 peuvent lire les nouveaux formats, vous pouvez migrer vos clients de chiffrement et de déchiffrement existants vers leurs versions V3 respectives.

Comprendre les concepts de la V3

La version 3 du client de chiffrement Amazon S3 introduit deux améliorations de sécurité clés : les politiques d'engagement et l'algorithme AES GCM with Key Commitment. Comprendre ces concepts est essentiel pour une migration réussie.

Politique d'engagement

Une politique d'engagement contrôle la manière dont le client de chiffrement gère l'engagement des clés lors des opérations de chiffrement et de déchiffrement. La version 3 propose trois options stratégiques :

`FORBID_ENCRYPT_ALLOW_DECRYPT`

Comportement de chiffrement : chiffre les objets sans engagement de clé.

Comportement de déchiffrement : permet le déchiffrement d'objets chiffrés avec ou sans clé d'engagement.

Implications en matière de sécurité : cette politique n'impose pas l'engagement des clés sur les objets récemment chiffrés, ce qui peut permettre la falsification des clés de données. Utilisez cette politique uniquement pendant la phase de migration initiale lorsque vous devez maintenir la compatibilité avec les clients V2.

Compatibilité des versions : les objets chiffrés avec cette politique peuvent être lus par toutes les implémentations V2 et V3.

`REQUIRE_ENCRYPT_ALLOW_DECRYPT`

Comportement de chiffrement : chiffre les objets avec un engagement clé à l'aide de l'`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`algorithme.

Comportement de déchiffrement : permet le déchiffrement d'objets chiffrés avec ou sans clé d'engagement.

Implications en matière de sécurité : cette politique renforce la sécurité des objets récemment chiffrés tout en préservant la capacité de lire les objets existants. Il s'agit de la politique recommandée pour la plupart des scénarios de migration.

Compatibilité des versions : les objets chiffrés avec cette politique ne peuvent être lus que par la version 3 et les dernières implémentations de la version 2.

Considérations relatives à la migration : Avant d'utiliser cette politique, assurez-vous que tous les clients devant lire les objets chiffrés ont été mis à niveau vers la version 3 ou la dernière version 2.

`REQUIRE_ENCRYPT_REQUIRE_DECRYPT`

Comportement de chiffrement : chiffre les objets avec un engagement clé à l'aide de l'`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`algorithme.

Comportement de déchiffrement : autorise uniquement le déchiffrement des objets chiffrés avec une clé d'engagement. Les objets chiffrés sans clé ne seront pas déchiffrés.

Implications en matière de sécurité : cette politique fournit le plus haut niveau de sécurité en imposant un engagement clé à la fois pour le chiffrement et le déchiffrement. Utilisez cette politique uniquement une fois que tous les objets ont été migrés afin d'utiliser l'engagement clé.

Compatibilité des versions : seules les implémentations V3 peuvent utiliser cette politique. Toute tentative de déchiffrement d'objets chiffrés V1 ou V2 avec cette politique échouera.

Considérations relatives à la migration : cette politique ne doit être utilisée qu'une fois la migration complète terminée et après avoir rechiffré tous les objets existants avec un engagement clé.

AES GCM avec un engagement clé

L'algorithme AES GCM with Key Commitment (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) est un nouvel algorithme de chiffrement introduit dans la version 3 qui fournit une protection contre les attaques de falsification des clés de données.

Amélioration de la sécurité : `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` protège contre la falsification des clés de données en liant cryptographiquement la clé de données au contenu crypté. Cela empêche les attaquants de substituer une clé de données différente lors du déchiffrement, ce qui pourrait entraîner le déchiffrement de données involontaires.

Compatibilité des versions : les objets chiffrés avec ne `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` peuvent être déchiffrés que par la version 3 et les dernières implémentations V2 du client de chiffrement Amazon S3. Les clients V1 ne peuvent pas déchiffrer les objets chiffrés avec cet algorithme.

> **Important**
>
> Exigence de mise à niveau : avant d'activer le chiffrement `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (en utilisant les politiques REQUIRE_ENCRYPT_ALLOW_DECRYPT ou REQUIRE_ENCRYPT_REQUIRE_DECRYPT), vous devez vous assurer que tous les clients devant lire les objets chiffrés ont été mis à niveau vers la version 3. Si tous les lecteurs ne sont pas mis à niveau, le déchiffrement des objets chiffrés avec clé d'engagement échouera.

Mettre à jour les clients existants pour lire les nouveaux formats

Le client de chiffrement V3 utilise des algorithmes de chiffrement et des fonctionnalités d'engagement clés que les anciennes versions du client ne prennent pas en charge. La première étape de la migration consiste à mettre à jour vos clients de déchiffrement V2 vers la dernière version du SDK. Une fois cette étape terminée, les clients V2 de votre application seront en mesure de déchiffrer les objets chiffrés par les clients de chiffrement V3. Consultez les détails ci-dessous pour chaque méthode d'installation du AWS SDK pour PHP.

Création et installation de la dernière version du SDK

Pour terminer cette migration, vous devez utiliser la dernière version du `aws/aws-sdk-php` package qui inclut le support du client de chiffrement V3.

Installation depuis Composer

Pour les projets installés à l'aide de Composer, dans le fichier Composer, mettez à jour le package du SDK avec la dernière version du SDK, puis exécutez la commande suivante.

```
composer update aws/aws-sdk-php
```

Installation à l'aide du fichier Phar ou Zip

Utilisez l’une des méthodes suivantes. Veillez à placer le fichier SDK mis à jour à l'emplacement requis par votre code, qui est déterminé par l'instruction require.

Pour les projets installés à l'aide du fichier Phar, téléchargez le fichier mis à jour : `aws.phar`.

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

Pour les projets installés à l'aide du fichier Zip, téléchargez le fichier mis à jour : .

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

Création, installation et déploiement d'applications

Après avoir mis à jour le SDK, reconstruisez et redéployez votre application pour vous assurer que tous les composants utilisent la version mise à jour. Cette étape est essentielle pour garantir que vos clients V2 peuvent lire les objets chiffrés par les clients V3.

Suivez les procédures de déploiement standard de votre entreprise pour déployer l'application mise à jour. Assurez-vous que toutes les instances de votre application sont mises à jour avant de procéder à la migration de vos clients de chiffrement et de déchiffrement vers la version 3.

Après le déploiement, vérifiez que votre application peut toujours déchiffrer les objets existants et qu'aucune erreur ne se produit lors des opérations normales. Cela confirme que la mise à jour du SDK a réussi et que votre application est prête pour la phase suivante de migration.

Migrer les clients de chiffrement et de déchiffrement vers la version 3

Après avoir mis vos clients à jour pour qu'ils lisent les nouveaux formats de chiffrement, vous pouvez mettre à jour vos applications pour les clients de chiffrement et de déchiffrement V3. Les exemples suivants vous montrent comment migrer avec succès votre code de la V2 à la V3.

Utilisation de clients de chiffrement V3

La V3 introduit la `S3EncryptionClientV3` classe et remplace `KmsMaterialsProviderV3` les équivalents V2. Les principales différences de la V3 sont les suivantes :

- La V3 utilise `KmsMaterialsProviderV3` (comme la V2) mais vérifie le contexte de chiffrement lors du déchiffrement d'objets dans les appels. `GetObject`
- La version 3 introduit des politiques d'engagement pour contrôler le comportement de chiffrement et de déchiffrement.

Exemple : migration de la V2 vers la V3 avec le chiffrement KMS

Pre-migration (V2)

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$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,
];

$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',
   '@CommitmentPolicy' => 'FORBID_ENCRYPT_ALLOW_DECRYPT',
   '@MaterialsProvider' => $materialsProvider,
   '@CipherOptions' => $cipherOptions,
   'Bucket' => $bucket,
   'Key' => $key,
]);
```

Pendant la migration (V3 avec rétrocompatibilité)

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
   new S3Client([
      'profile' => 'default',
      'region' => 'us-east-1',
      'version' => 'latest',
   ])
);

// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
   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,
   '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
   '@KmsEncryptionContext' => ['context-key' => 'context-value'],
   'Bucket' => $bucket,
   'Key' => $key,
   'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
```

```
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Post-migration (V3 avec engagement clé)

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    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,
    // Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
    // This encrypts with key commitment and does not decrypt V2 objects
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3',
    // Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
    // This encrypts with key commitment and does not decrypt V2 objects
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Principales différences dans la V3 :

- Utilisez `KmsMaterialsProviderV3` au lieu de `KmsMaterialsProviderV2`
- Le `@KmsEncryptionContext` paramètre est toujours requis pour les `putObject` opérations
- Le `@KmsEncryptionContext` paramètre est facultatif pour les `getObject` opérations et permet de vérifier que le contexte de chiffrement fourni correspond à celui de l'objet.
- Le `@SecurityProfile` paramètre contrôle les versions de chiffrement qui peuvent être déchiffrées. Paramétré sur `'V3_AND_LEGACY'` pour prendre en charge la lecture des objets chiffrés V1 et V2 pendant la migration
- Le `@CommitmentPolicy` paramètre contrôle la politique d'engagement pour cette opération. Paramétré `'FORBID_ENCRYPT_ALLOW_DECRYPT'` sur pour permettre la lecture d'objets chiffrés sans engagement pendant la migration

Exemples supplémentaires

Les exemples suivants illustrent les options de configuration supplémentaires disponibles dans la version 3 qui peuvent vous aider à gérer le processus de migration et à contrôler le comportement de chiffrement.

Activation de la prise en charge de hérité

Au cours de la migration, vous devrez peut-être déchiffrer des objets chiffrés avec les versions V1 ou V2 du client de chiffrement Amazon S3. Le `@SecurityProfile` paramètre contrôle les versions de chiffrement que votre client V3 peut déchiffrer.

Quand utiliser cette configuration : utilisez le profil de `'V3_AND_LEGACY'` sécurité lorsque votre application doit lire des objets chiffrés par des clients V1 ou V2. Cela est courant pendant la période de migration lorsque vous avez un mélange d'anciens et de nouveaux objets chiffrés dans vos buckets.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
```

```
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

// Decrypt objects encrypted with V1, V2, or V3
$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Le paramètre `@SecurityProfile` accepte les valeurs suivantes :

- `'V3'` (par défaut) : décryptez uniquement les objets chiffrés avec V3 en utilisant un engagement par clé
- `'V3_AND_LEGACY'` : déchiffrez les objets chiffrés avec V1, V2 ou V3

> **Important**
>
> Après avoir terminé votre migration et rechiffré tous les objets avec la version 3, vous devez supprimer le `@SecurityProfile` paramètre ou le définir `'V3'` pour garantir une sécurité maximale.

Configuration de la méthode de stockage

Le client de chiffrement Amazon S3 peut stocker les métadonnées de chiffrement de deux manières : dans les en-têtes de métadonnées de l'objet ou dans un fichier d'instructions distinct. Le `@MetadataStrategy` paramètre contrôle la méthode de stockage utilisée.

Quand utiliser cette configuration : Utilisez-la `'INSTRUCTION_FILE'` lorsque vous devez conserver les métadonnées d'origine de l'objet ou lorsque vous travaillez avec des objets soumis à des contraintes de taille de métadonnées. À utiliser `'METADATA'` (par défaut) pour des déploiements plus simples où les métadonnées de chiffrement peuvent être stockées à côté de l'objet.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
```

```
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

// Store encryption metadata in a separate instruction file
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'INSTRUCTION_FILE',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

// Store encryption metadata in object headers (default)
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
```

```
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'METADATA',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

Le paramètre `@MetadataStrategy` accepte les valeurs suivantes :

- `'METADATA'`(par défaut) : stockez les métadonnées de chiffrement dans les en-têtes de métadonnées de l'objet
- `'INSTRUCTION_FILE'`: Stockez les métadonnées de chiffrement dans un fichier d'instructions distinct avec le suffixe `.instruction`

> **Note**
>
> Lors de son utilisation`'INSTRUCTION_FILE'`, l'algorithme AES GCM with Key Commitment fournit une protection supplémentaire contre la falsification des clés de données. Les objets utilisant le `'METADATA'` stockage ne bénéficient pas de cette protection supplémentaire.

FAQ pour AWS SDK pour PHP la version 3

Quelles méthodes sont disponibles sur un client ?

Il AWS SDK pour PHP utilise des descriptions de service et des méthodes magiques dynamiques __call () pour exécuter des opérations d'API. Vous trouverez la liste complète des méthodes disponibles pour un client de service Web dans la documentation d'API du client.

Que dois-je faire en cas d'erreur de certificat cURL SSL ?

Ce problème peut se produire lors de l'utilisation d'un bundle out-of-date CA avec cURL et SSL. Vous pouvez contourner ce problème en mettant à jour le bundle CA sur votre serveur ou en téléchargeant un autre bundle up-to-date CA directement depuis le site Web cURL.

Par défaut, ils AWS SDK pour PHP utiliseront le bundle CA configuré lors de la compilation de PHP. Vous pouvez modifier le bundle d'autorité de certification (CA) par défaut utilisé par PHP en modifiant le paramètre de configuration .ini PHP `openssl.cafile` à définir par le chemin d'accès d'un fichier d'autorité de certification (CA) sur disque.

Quelles versions d'API sont disponibles pour un client ?

Une option `version` est requise lors de la création d'un client. Une liste des versions d'API disponibles se trouve sur la page de documentation de l'API de chaque client : : aws-php-class :<index.html>. Si vous ne parvenez pas à charger une version d'API spécifique, il se peut que vous deviez mettre à jour votre copie du kit AWS SDK pour PHP.

Vous pouvez spécifier la chaîne `latest` dans la valeur de configuration « version » pour utiliser la version d'API disponible la plus récente du fournisseur d'API de votre client (le fournisseur d'API par défaut recherchera les modèles d'API disponibles dans le répertoire `src/data` du kit SDK).

> **Warning**
>
> Nous vous déconseillons de spécifier `latest` dans une application en production. En effet, l'intégration d'une nouvelle version mineure du kit SDK comprenant une mise à jour d'API peut entraîner l'échec de votre application en production.

Quelles versions de région sont disponibles pour un client ?

L'option `region` est requise lors de la création d'un client et doit être spécifiée à l'aide d'une valeur de chaîne. Pour obtenir la liste des AWS régions et points de terminaison disponibles, consultez la section AWS Régions et points de terminaison dans le. Références générales AWS

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

Pourquoi ne puis-je pas charger ou télécharger des fichiers de plus de 2 Go ?

Le type entier de PHP étant signé et de nombreuses plateformes utilisant les entiers 32 bits, le kit AWS SDK pour PHP ne gère pas correctement les fichiers de plus de 2 Go sur une pile 32 bits (où « pile » inclut l'UC, le système d'exploitation, le serveur Web et le binaire PHP). Il s'agit d'un problème connu de PHP. Dans le cas de Microsoft Windows, seules les versions de PHP 7 prennent en charge les entiers 64 bits.

La solution recommandée consiste à utiliser une pile Linux 64 bits, telle que l'AMI Linux Amazon 64 bits, dotée de la dernière version de PHP.

Pour plus d'informations, consultez la section Taille de fichier PHP : valeurs renvoyées.

Comment savoir quelles données sont envoyées sur le réseau ?

Vous pouvez obtenir des informations de débogage, y compris les données envoyées sur le réseau, à l'aide de l'option `debug` d'un constructeur client. Lorsque cette option est définie sur `true`, l'ensemble des mutations de la commande exécutée, de la requête envoyée, de la réponse reçue et du résultat traité sont émises vers STDOUT. Cela inclut les données envoyées et reçues sur le réseau.

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

```
    'debug'   => true
]);
```

Comment puis-je définir des en-têtes arbitraires sur une requête ?

Vous pouvez ajouter des en-têtes arbitraires à une opération de service en ajoutant un intergiciel personnalisé à la `Aws\HandlerList` d'une `Aws\CommandInterface` ou d'une `Aws\ClientInterface`. L'exemple suivant montre comment ajouter un `X-Foo-Baz` en-tête à une `PutObject` opération Amazon S3 spécifique à l'aide de la méthode d'`Aws\Middleware::mapRequest`assistance.

Pour plus d'informations, consultez la section mapRequest.

Comment puis-je signer une requête arbitraire ?

Vous pouvez signer une requête arbitraire : aws-php-class : PSR-7 <class-PSR.http.Message.RequestInterface.html> en utilisant la classe : aws-php-class : SignatureV4 du SDK. <class-Aws.Signature.SignatureV4.html>

Consultez Signing Custom CloudSearch Domain Requests with Amazon Domain Requests with AWS SDK pour PHP Version 3 pour un exemple complet de la procédure à suivre.

Comment puis-je modifier une commande avant de l'envoyer ?

Vous pouvez modifier une commande avant de l'envoyer en ajoutant un intergiciel personnalisé à la `Aws\HandlerList` d'une `Aws\CommandInterface` ou d'une `Aws\ClientInterface`. L'exemple suivant montre comment ajouter des paramètres de commande personnalisés à une commande avant de l'envoyer, principalement en ajoutant des options par défaut. Cet exemple utilise la méthode d'assistance `Aws\Middleware::mapCommand`.

Pour plus d'informations, consultez la section mapCommand.

Qu'est-ce qu'un CredentialsException ?

Si vous constatez `Aws\Exception\CredentialsException` que vous utilisez le AWS SDK pour PHP, cela signifie que le SDK n'a reçu aucune information d'identification et qu'il n'a pas pu trouver d'informations d'identification dans l'environnement.

Si vous instanciez un client sans information d'identification, le kit SDK tente de trouver des informations d'identification à la première exécution d'une opération de service. Il intègre d'abord certaines variables d'environnement spécifiques, puis il recherche les informations d'identification du profil d'instance, qui ne sont disponibles que sur les EC2 instances Amazon configurées. En l'absence d'information d'identification, une `Aws\Exception\CredentialsException` est levée.

Si cette erreur s'affiche et que vous avez l'intention d'utiliser les informations d'identification du profil d'instance, vous devez vous assurer que l' EC2 instance Amazon sur laquelle le SDK est exécuté est configurée avec un rôle IAM approprié.

Si vous rencontrez cette erreur et que vous ne prévoyez pas d'utiliser les informations d'identification du profil d'instance, veillez à fournir des informations d'identification correctes au kit SDK.

Pour plus d'informations, consultez la section Informations d'identification pour la AWS SDK pour PHP version 3.

Est-ce que cela AWS SDK pour PHP fonctionne sur HHVM ?

Le AWS SDK pour PHP ne fonctionne pas actuellement sur HHVM et ne pourra pas le faire tant que le problème lié à la sémantique du rendement dans HHVM ne sera pas résolu.

Comment désactiver SSL ?

Vous pouvez désactiver SSL en définissant le paramètre `scheme` d'une méthode de fabrique de client sur « http ». Il est important de noter que tous les services ne prennent pas en charge l'accès `http`. Voir AWS Régions et points de terminaison dans le Références générales AWS pour obtenir la liste des régions, des points de terminaison et des schémas pris en charge.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

> **Warning**
>
> SSL impose que toutes les données soient chiffrées et requiert davantage de paquets TCP que le protocole TCP seul pour établir une liaison de connexion. La désactivation de SSL

peut donc se traduire par une légère amélioration des performances. Cependant, une fois SSL désactivé, toutes les données sont envoyées sur le réseau non chiffrées. Avant de désactiver SSL, prenez en compte les répercussions de sa désactivation sur la sécurité et le risque d'écoute illicite sur le réseau.

Que dois-je faire face à une « erreur d'analyse » ?

Le moteur PHP génère des erreurs d'analyse lorsqu'il rencontre une syntaxe qu'il ne comprend pas. Cette situation se produit quasi systématiquement lorsque le moteur tente d'exécuter du code écrit pour une autre version de PHP.

Si vous rencontrez une erreur d'analyse, vérifiez votre système et assurez-vous qu'il répond aux exigences et aux recommandations du SDK pour la AWS SDK pour PHP version 3.

Pourquoi le client Amazon S3 décompresse-t-il les fichiers compressés au format gzippé ?

Certains gestionnaires HTTP, dont le gestionnaire HTTP Guzzle 6 par défaut, gonflent le corps des réponses compressées par défaut. Vous pouvez remplacer ce comportement en définissant l'option HTTP decode_content sur `false`. Pour des raisons de rétrocompatibilité, cette valeur par défaut ne peut pas être modifiée, mais nous vous recommandons de désactiver le décodage du contenu au niveau du client S3.

Consultez la section decode_content pour obtenir un exemple de désactivation du décodage de contenu automatique.

Comment désactiver la signature corporelle dans Amazon S3 ?

Vous pouvez désactiver la signature du corps en définissant le paramètre `ContentSHA256` de l'objet de commande sur `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`. Ensuite, ils l' AWS SDK pour PHP utiliseront comme en-tête « x-amz-content-sha -256 » et comme somme de contrôle du corps de la demande canonique.

```
$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);
```

Comment le schéma de nouvelle tentative est-il géré dans le kit AWS SDK pour PHP ?

AWS SDK pour PHP Possède un `RetryMiddleware` qui gère le comportement des nouvelles tentatives. En ce qui concerne les codes de statut HTTP 5xx pour les erreurs de serveur, le kit SDK lance une nouvelle tentative sur 500, 502, 503 et 504.

Les exceptions de limitation, y compris `RequestLimitExceeded`, `Throttling`, `ProvisionedThroughputExceededException`, `ThrottlingException`, `RequestThrottled` et `BandwidthLimitExceeded`, sont également gérées avec les nouvelles tentatives.

AWS SDK pour PHP Il intègre également un délai exponentiel à un algorithme de recul et de gigue dans le schéma de réessai. En outre, le comportement de nouvelle tentative par défaut est configuré comme `3` pour tous les services, à l'exception d'Amazon DynamoDB, qui l'est. `10`

Comment gérer les exceptions avec des codes d'erreur ?

Outre `Exception` les classes AWS SDK pour PHP-custom, chaque client AWS de service possède sa propre classe d'exception qui hérite de. AwsExceptionAwsException Vous pouvez affiner les types d'erreur à identifier avec les erreurs d'API spécifiques répertoriées dans la section `Errors` de chaque méthode.

Les informations relatives au code d'erreur sont disponibles avec getAwsErrorCode () à partir de`Aws\Exception\AwsException`.

```
$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;
    }
}
```

Glossaire

Version de l'API

Les services disposent d'une ou de plusieurs versions d'API et la version que vous utilisez détermine les opérations et les paramètres valides. Le format des versions d'API est similaire à celui d'une date. Par exemple, la dernière version de l'API pour Amazon S3 est`2006-03-01`. Spécifiez une version lorsque vous configurez un objet client.

Client

Les objets client permettent d'exécuter des opérations pour un service. Chaque service pris en charge dans le kit SDK est associé à un objet client. Les objets clients ont des méthodes qui correspondent one-to-one aux opérations de service. Consultez le the section called “Création d'un client de service de base” pour commencer à utiliser les clients.

Command

Les objets de commande encapsulent l'exécution d'une opération. Lorsque vous exécutez une opération de service telle que décrite dans la section relative à la demande, vous ne traitez pas directement des objets de commande. Vous pouvez accéder aux objets de commande à l'aide de la méthode `getCommand()` d'un client afin de pouvoir utiliser les fonctions avancées du kit SDK, telles que les demandes simultanées et le traitement par lots. Consultez les objets de commande dans le guide de la AWS SDK pour PHP version 3 pour plus de détails.

Handler (Gestionnaire)

Un gestionnaire est une fonction qui transforme une commande et une requête en un résultat. Il envoie généralement des requêtes HTTP. Les gestionnaires peuvent comporter des intergiciels, qui renforcent leur comportement. Un gestionnaire est une fonction qui accepte un `Aws\CommandInterface` et un `Psr\Http\Message\RequestInterface`, et renvoie une promesse exécutée avec un `Aws\ResultInterface` ou rejetée avec une raison `Aws\Exception\AwsException`.

JMESPath

JMESPathest un langage de requête pour les données de type JSON. AWS SDK pour PHP Utilise des JMESPath expressions pour interroger les structures de données PHP. JMESPath les expressions peuvent être utilisées directement sur `Aws\Result` des `Aws\ResultPaginator` objets via la `search($expression)` méthode.

Intergiciel

Un intergiciel est un type spécial de fonction de haut niveau qui augmente le comportement de transfert d'une commande et délègue la tâche à un gestionnaire « suivant ». Les fonctions d'intergiciel acceptent un `Aws\CommandInterface` et un `Psr\Http\Message\RequestInterface`, et renvoient une promesse exécutée avec un `Aws\ResultInterface` ou rejetée avec une raison `Aws\Exception\AwsException`.

Opération

Fait référence à une seule opération au sein de l'API d'un service (par exemple, `CreateTable` pour DynamoDB, `RunInstances` pour Amazon). EC2 Dans le kit SDK, les opérations sont exécutées en appelant une méthode du même nom sur l'objet client du service correspondant. L'exécution d'une opération implique de préparer et d'envoyer une demande HTTP au service, puis d'en analyser la réponse. Ce processus d'exécution d'une opération est extrait par le kit SDK via les objets de commande.

Programme de pagination

Certaines opérations AWS de service sont paginées et donnent des résultats tronqués. Par exemple, l'`ListObjects`opération d'Amazon S3 ne renvoie que 1 000 objets à la fois. De telles opérations nécessitent d'effectuer des requêtes ultérieures avec des paramètres de jeton (ou de marqueur) pour récupérer l'ensemble complet des résultats. Les paginateurs sont une fonctionnalité du SDK qui agit comme une abstraction de ce processus afin de faciliter l'utilisation de la pagination par les développeurs. APIs Ils sont accessibles via la méthode `getPaginator()` du client. Consultez les paginateurs dans le guide de la AWS SDK pour PHP version 3 pour plus de détails.

Promesse

Une promesse représente le résultat final d'une opération asynchrone. Le principal moyen d'interagir avec une promesse réside dans l'utilisation de sa méthode then, qui enregistre les rappels pour recevoir la valeur finale d'une promesse ou la raison pour laquelle la promesse ne peut pas être exécutée.

Région

Les services sont pris en charge dans une ou plusieurs régions géographiques. Les services peuvent être différents endpoints/URLs dans chaque région, ce qui permet de réduire la latence des données dans vos applications. Fournissez une région lors de la configuration d'un objet client, pour permettre au kit SDK de déterminer le point de terminaison à utiliser avec le service.

SDK

Le terme « SDK » peut faire référence à la AWS SDK pour PHP bibliothèque dans son ensemble, mais également à la `Aws\Sdk` classe (docs), qui agit comme une fabrique pour les objets clients pour chaque service. La classe `Sdk` vous permet également de fournir un ensemble de valeurs de configuration globale appliquées à tous les objets clients qu'elle crée.

Service

Une manière générale de faire référence à l'un des AWS services (par exemple, Amazon S3, Amazon DynamoDB AWS OpsWorks, etc.). Chaque service est associé, dans le kit SDK, à un objet client prenant en charge une ou plusieurs versions d'API. Chaque service dispose également d'une ou de plusieurs opérations qui composent son API. Les services sont pris en charge dans une ou plusieurs régions.

Signature

Lors de l'exécution d'opérations, le kit SDK utilise vos informations d'identification pour créer une signature numérique de votre demande. Le service vérifie ensuite la signature avant de traiter votre demande. Le processus de signature est encapsulé par le kit SDK et s'effectue automatiquement à l'aide des informations d'identification configurées pour le client.

Programme d'attente

Les programmes d'attente sont une fonction du kit SDK permettant de faciliter l'utilisation des opérations modifiant l'état d'une ressource. Ces programmes sont cohérents à terme ou de nature asynchrone. Par exemple, l'opération Amazon `CreateTable` DynamoDB renvoie immédiatement une réponse, mais il se peut que la table ne soit pas prête à être consultée avant plusieurs secondes. L'exécution d'un programme d'attente vous permet d'attendre qu'une ressource passe à un état spécifique en mettant en veille et en interrogeant l'état de la ressource. Les programmes d'attente sont accessibles via la méthode `waitUntil()` du client. Consultez le guide Waiters in the AWS SDK pour PHP Version 3 pour plus de détails.

Pour consulter la AWS terminologie la plus récente, consultez le AWS glossaire dans le Références générales AWS.

Historique du document

Les tableaux suivants décrivent les modifications importantes apportées depuis la dernière version du AWS SDK pour PHP Developer Guide.

Changements les plus récents :

Modification	Description	Date
Mises à jour de la table des matières	La table des matières a été révisée pour être plus cohérente avec les autres guides du SDK.	28 juillet 2025
Protection de l'intégrité des données grâce à des checksums	Contenu mis à jour avec des détails sur le calcul automatiq ue de la somme de contrôle.	16 janvier 2025
Révisions des sujets relatifs aux informations	Sujet réorganisé. Plus de détails sont fournis pour la chaîne de fournisseurs d'informations d'identification par défaut.	10 janvier 2025
Migration du client de chiffrement Amazon S3 vers la version V3	Ajout d'un sujet sur la migration du client de chiffrement Amazon S3 de la V2 vers la V3	4 décembre 2024
Chargements partitionnés sur Amazon S3	Documentez le tableau « params » qui peut être utilisé pour configurer les sous-commandes de et `ObjectUpl oader` `Multipart Uploader`	6 novembre 2024
Téléchargeur d'objets	Clarifier l'utilisation des rappels disponibles sur les télécharg	11 octobre 2024

	ements `ObjectUploader` pour S3	
Mettre à jour les noms des compartiments Amazon S3	Les noms des compartiments S3 ont été mis à jour tout au long du guide.	30 septembre 2024
Points de terminaison EventBridge mondiaux Amazon	Ajouter un exemple de code qui montre comment utiliser les points de terminaison EventBridge mondiaux Amazon	22 décembre 2023
AWS Temps d'exécution commun (AWS CRT)	Ajoutez une rubrique qui traite de l'utilisation du AWS Common Runtime (AWS CRT) par le SDK pour PHP.	17 novembre 2023
StreamWrapper mises à jour de mkdir ()	Ajoutez des informations sur l'utilisation des compartim ents et des objets de dossier à l'aide `mkdir()` de.	2 novembre 2023
Création d'un client de service	Mettez à jour les extraits de code en supprimant le paramètre « version », car la valeur par défaut est « la plus récente ».	31 août 2023
Création de clients de base	Mettez à jour les extraits de code en supprimant le paramètre « version », car la valeur par défaut est « la plus récente ».	31 août 2023
Table des matières	Table des matières mise à jour pour rendre les exemples de code plus accessibles.	1er juin 2023

Mises à jour des bonnes pratiques IAM	Mise à jour du guide s'aligner sur les bonnes pratiques IAM. Pour plus d'informations, consultez Bonnes pratiques de sécurité dans IAM. Mises à jour de Getting started.	20 mai 2023
Chargements partitionnés sur Amazon S3	Informations de configuration incluses pour les télécharg ements synchrones. Ajout de l'option de `add_conte nt_md5` téléchargement pour les téléchargements asynchrones.	13 avril 2023
Transferts d'annuaires Amazon S3	L'option de `add_conte nt_md5` transfert a été ajoutée.	13 avril 2023
Informations de référence	Ajout de plusieurs liens vers du contenu détaillé pertinent dans le Guide de référence AWS des kits de développe ment logiciel et des outils. Mise à jour du formatage du guide.	14 septembre 2022
Nettoyage général	Ajout de références au Guide de référence AWS des kits de développement logiciel et des outils. AWS Key Management Service Sections mises à jour pour refléter les mises à jour terminologiques.	23 août 2022
Travailler avec les AWS services	Liste incluse des exemples de code disponibles sur GitHub.	1er avril 2022

Activation des métriques du SDK	Suppression des informations relatives à l'activation des métriques du SDK, devenues obsolètes le 20 décembre 2021.	27 janvier 2022
Migration du client de chiffrement Amazon S3	Ajout d'un sujet sur la migration du client de chiffrement Amazon S3	7 août 2020

Anciennes modifications :

Modifier	Description	Date de publication
Exemples de Secrets Manager	Ajout d'autres exemples de service	27 mars 2019
Découverte du points de terminaison	Configuration de la découverte de points de terminaison	15 février 2019
Amazon CloudFront	Ajout d'autres exemples de service	25 janvier 2019
Fonctions de service	Métriques SDK	11 janvier 2018
Amazon Kinesis, Amazon SNS	Ajout d'autres exemples de service	14 décembre 2018
Exemples Amazon SES	Ajout d'autres exemples de service	5 octobre 2018
AWS KMS exemples	Ajout d'autres exemples de service	8 août 2018
Informations d’identification	Clarification et simplification du guide des informations d'identification	30 juin 2018

Modifier	Description	Date de publication
MediaConvert exemples	Ajout d'autres exemples de service	15 juin 2018
Nouvelle présentation Web	La documentation est passée au AWS style	9 mai 2018
Chiffrement Amazon S3	Chiffrement côté client	17 novembre 2017
Amazon S3, Amazon SQS	Ajout d'autres exemples de service	26 mars 2017
Amazon S3, IAM, Amazon EC2	Ajout d'autres exemples de service	17 mars 2017
Ajout d'informations d'identif ication	Ajoute le support pour AssumeRole et ini	17 janvier 2017
Exemples S3	S3 Multi-Region et publicati ons présignées	18 mars 2016
OpenSearch Service et Amazon CloudSearch	Ajout d'autres exemples de service	28 décembre 2015
Ligne de commande	Ajout de paramètres de commande	13 août 2015
Fonctions de service	Ajoute des fonctionnalités de service pour S3 et AWS	30 avril 2015
Nouvelle version SDK	Version 3 de la version AWS SDK pour PHP publiée.	26 mai 2015

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.