개발자 가이드 버전 3

AWS SDK for PHP

AWS SDK for PHP: 개발자 가이드 버전 3

Table of Contents

AWS SDK for PHP 버전 3이란 무엇인가요?

AWS SDK for PHP 버전 3를 통해 PHP 개발자는 PHP 코드에서 Amazon Web Services를 사용하고 Amazon S3, Amazon DynamoDB, Glacier, Amazon Glacier 등의 서비스를 사용하여 강력한 애플리케이션과 소프트웨어를 구축할 수 있습니다. Composer를 통해 SDK를 설치하거나 `aws/aws-sdk-php` 패키지를 요구하거나 독립 실행형 `aws.zip` 또는 `aws.phar` 파일을 다운로드하여 몇 분 이내에 시작할 수 있습니다.

일부 서비스는 SDK에서 즉시 사용할 수 없습니다. AWS SDK for PHP에서 현재 지원되는 서비스를 찾아보는 방법은 서비스 이름 및 API 버전을 참조하십시오.

> **Note**
>
> SDK 버전 2를 사용하는 코드를 버전 3으로 마이그레이션하려면 AWS SDK for PHP 버전 2에서 업그레이드를 읽어 보세요.

SDK 시작하기

SDK를 직접 사용해 볼 준비가 되었다면 이 AWS SDK for PHP 버전 3 시작하기 장을 참고하세요. AWS를 사용하여 인증하고, 개발 환경을 설정하고, Amazon S3를 사용하여 첫 번째 기본 애플리케이션을 생성하는 과정을 안내합니다.

추가 리소스

- FAQ
- 용어집
- AWS SDK 및 도구 참조 안내서: AWS SDK에서 흔히 사용되는 설정, 기능 및 기타 기본 개념이 포함되어 있습니다.
- Guzzle 설명서
- AWS SDK for PHP를 사용하는 코드 예제는 awsdocs/aws-doc-sdk-examples 리포지토리에서 확인할 수 있습니다.
- Gitter의 PHP SDK 커뮤니티.
- AWS re:Post.

GitHub:

- AWS SDK for PHP의 소스 코드는 aws/aws-sdk-php 리포지토리에서 확인할 수 있습니다.
- SDK에 기여
- 버그 보고 또는 기능 요청

API 설명서

SDK에 대한 API 설명서는 https://docs.aws.amazon.com/sdk-for-php/latest/reference/에서 확인할 수 있습니다.

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 메이저 버전 및 기본 종속성의 유지 관리 및 지원에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 다음 내용을 참조하세요.

- AWS SDK 및 도구 유지 관리 정책
- AWS SDK 및 도구 버전 지원 매트릭스

AWS SDK for PHP 버전 3 시작하기

이 장에서는 AWS SDK for PHP 버전 3을 준비하고 실행하는 과정만 다룹니다.

주제

AWS SDK for PHP 버전 3에 대한 요구 사항 및 권장 사항

AWS SDK for PHP에서 최적의 결과를 얻으려면 사용 중인 환경이 다음 요구 사항 및 권장 사항을 지원해야 합니다.

요구 사항

AWS SDK for PHP를 사용하려면 SimpleXML PHP 확장이 활성화된 PHP 버전 8.1 이상을 사용해야 합니다. 프라이빗 Amazon CloudFront URL에 서명해야 하는 경우 OpenSSL PHP 확장도 필요합니다.

추천

최소 요건에 더해, 다음을 설치, 제거, 사용하는 것이 좋습니다.

cURL 7.16.2 이상 설치	OpenSSL/NSS 및 zlib로 컴파일된 최신 버전의 cURL을 사용합니다. cURL이 시스템에 설치되어 있지 않고 클라이언트에 대한 사용자 지정 http_handler를 구성하지 않은 경우 SDK에서는 PHP 시스템 래퍼를 사용합니다.
OPCache 사용	공유 메모리에 미리 컴파일된 스크립트 바이트코드를 저장하여 PHP 성능을 개선하려면 OPcache 확장을 사용합니다. 그러면 PHP에서 각 요청에 대해 스크립트를 로드하여 구문 분석

	할 필요가 없습니다. 이 확장은 기본적으로 활성화됩니다. Amazon Linux를 실행할 경우 OPCache 확장을 사용하려면 php56-opcache 또는 php55-opcache yum 패키지를 설치해야 합니다.
프로덕션 환경에서 Xdebug 제거	Xdebug를 사용하면 성능 병목 현상을 파악할 수 있습니다. 하지만 성능이 애플리케이션에 중요한 경우 Xdebug 확장을 프로덕션 환경에 설치하지 마세요. 확장을 로드하면 SDK 성능이 매우 느려집니다.
Composer 클래스맵 자동 로더 사용	자동 로더는 PHP 스크립트에 요구된 클래스를 로드합니다. Composer는 AWS SDK for PHP를 비롯하여 애플리케이션의 PHP 스크립트와 애플리케이션에 필요한 모든 다른 PHP 스크립트를 자동으로 로드할 수 있는 자동 로더를 생성합니다. 프로덕션 환경에서는 클래스맵 자동 로더를 사용하여 자동 로더 성능을 개선하는 것이 좋습니다. `-o` 또는 `==optimize-autoloader` 옵션을 Composer의 설치 명령에 전달하여 클래스맵 자동 로더를 생성할 수 있습니다.

호환성 테스트

SDK 코드 베이스에 위치한 `compatibility-test.php` 파일을 실행하여 시스템에서 SDK를 실행할 수 있는지 확인합니다. SDK의 최소 시스템 요구 사항을 충족하는 것 외에도 호환성 테스트에서는 선택적 설정을 검사하고 성능을 개선할 수 있는 권장 사항을 제공합니다. 호환성 테스트 결과는 명령줄 또는 웹 브라우저에 출력됩니다. 브라우저에서 테스트 결과를 검토할 경우 성공적인 검사는 녹색, 경고는 보라색, 실패는 빨간색으로 표시됩니다. 명령줄에서 실행할 경우 검사 결과가 별도의 줄에 표시됩니다.

SDK에서 문제를 보고할 때 호환성 테스트 출력을 공유하면 근본적인 이유를 파악하는 데 도움이 됩니다.

AWS SDK for PHP 버전 3 설치

다음 방법으로 AWS SDK for PHP 버전 3을 설치할 수 있습니다.

- Composer를 통한 종속 항목 이용
- 사전 패키지된 SDK의 phar 이용
- SDK의 ZIP 파일 이용

AWS SDK for PHP 버전 3을 설치하기 전에 해당 환경에서 PHP 버전 8.1 이상을 사용 중인지 확인하세요. 환경의 요구 사항 및 권장 사항에 대해 자세히 알아봅니다.

> **Note**
>
> .phar 및.zip 메서드를 통해 SDK를 설치하려면 Multibyte String PHP 확장을 별도로 설치하고 활성화해야 합니다.

Composer를 통한 종속 항목으로 AWS SDK for PHP 설치

Composer를 통해 AWS SDK for PHP를 설치하는 것이 좋습니다. Composer는 프로젝트의 종속 항목을 관리 및 설치하는 PHP용 도구입니다.

Composer를 설치하고, 자동 로딩을 구성하고, 각종 모범 사례에 따라 종속 항목을 정의하는 방법에 대한 자세한 내용은 getcomposer.org를 참조하세요.

Composer 설치

프로젝트에 Composer가 없는 경우, Composer 다운로드 페이지에서 Composer를 다운로드 및 설치하세요.

- Windows의 경우, Windows 설치 프로그램 지침을 따르세요.
- Linux의 경우 명령줄 설치 지침을 따르세요.

Composer를 통해 종속 항목으로 AWS SDK for PHP 추가

시스템에 Composer가 이미 전역 설치되어 있다면 프로젝트의 기본 디렉터리에서 다음을 실행하여 AWS SDK for PHP를 종속 항목으로 설치합니다.

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

그렇지 않으면 이 Composer 명령을 입력하여 AWS SDK for PHP 최신 버전을 종속 항목으로 설치합니다.

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

php 스크립트에 자동 로더 추가

Composer 설치 작업은 사용자 환경에 여러 폴더와 파일을 만듭니다. 사용할 주 파일은 `autoload.php`이고 환경의 `vendor` 폴더에 있습니다.

스크립트에서 AWS SDK for PHP를 활용하려면 다음과 같이 스크립트에 자동 로더를 포함시켜야 합니다.

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

패키지드 Phar를 사용하여 설치

각 AWS SDK for PHP 릴리스에는 SDK 실행에 필요한 모든 클래스와 종속 항목을 담은 사전 패키지된 phar(PHP 아카이브)가 포함되어 있습니다. 또한 이 phar는 AWS SDK for PHP 및 모든 종속 항목을 위한 클래스 자동 로더를 자동으로 등록합니다.

패키지된 phar를 다운로드하여 스크립트에 포함시킬 수 있습니다.

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

> **Note**
>
> Suhosin 패치를 적용한 PHP는 사용하지 않는 것이 좋지만, Ubuntu 및 Debian 배포에서는 일반적으로 사용됩니다. 이 경우 suhosin.ini에서 phar 사용을 활성화해야 할 수도 있습니다. 그렇지 않을 경우 코드에 phar 파일을 포함하면 자동으로 실패합니다. suhosin.ini를 수정하려면 다음 줄을 추가합니다.

```
suhosin.executor.include.whitelist = phar
```

ZIP 파일을 사용하여 설치

AWS SDK for PHP에는 SDK를 실행하는 데 필요한 모든 클래스와 종속 항목을 묶은 ZIP 파일이 포함되어 있습니다. 또한 이 ZIP 파일에는 AWS SDK for PHP 및 종속 항목을 위한 클래스 자동 로더도 들어 있습니다.

SDK를 설치하려면 .zip 파일을 다운로드한 다음 선택한 위치에 프로젝트로 풀어 놓습니다. 그런 다음 아래와 같이 스크립트에 자동 로더를 포함시킵니다.

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

AWS SDK for PHP 버전 3을 AWS 사용하여 로 인증

를 사용하여 개발할 AWS 때 코드가를 인증하는 방법을 설정해야 합니다 AWS 서비스. 환경 및 사용 가능한 액세스에 따라 다양한 방식으로 AWS 리소스에 대한 프로그래밍 방식 AWS 액세스를 구성할 수 있습니다.

인증 방법을 선택하고 SDK에 맞게 구성하려면AWS SDK 및 도구 참조 안내서의 Authentication and access를 참조하세요.

로컬 개발용 IAM Identity Center 인증 설정

로컬에서 개발 중이고 고용주로부터 인증 방법을 받지 않은 신규 사용자는 설정해야 합니다 AWS IAM Identity Center. 이 방법에는 구성이 쉽고 AWS 액세스 포털 AWS CLI 에 정기적으로 로그인하기 위한 설치가 포함됩니다. 이 방법을 선택하는 경우AWS SDK 및 도구 참조 안내서의 IAM Identity Center authentication 절차를 완료한 후 환경에 다음 요소가 포함되어야 합니다.

- 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 AWS CLI사용하는 입니다.
- SDK에서 참조할 수 있는 구성 값 세트가 포함된 `[default]` 프로필이 있는 공유 AWSconfig 파일입니다. 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 공유 파일의 위치를 참조하세요.

- 공유 `config` 파일에는 `region` 설정이 포함됩니다. 이렇게 하면 SDK AWS 리전 가 요청에 사용하는 기본값이 설정됩니다. 이 리전은 `region` 속성으로 명시적으로 구성되지 않은 SDK 서비스 요청에 사용됩니다.
- SDK는 AWS에 요청을 보내기 전에 프로필의 SSO token provider configuration을 사용하여 보안 인증을 얻습니다. IAM Identity Center 권한 세트에 연결된 IAM 역할인 `sso_role_name` 값은 애플리케이션에 AWS 서비스 사용되는에 대한 액세스를 허용합니다.

  다음 샘플 `config` 파일은 SSO 토큰 공급자 구성으로 설정된 기본 프로필을 보여줍니다. 프로필의 `sso_session` 설정은 이름이 지정된 `sso-session` section을 참조합니다. `sso-session` 섹션에는 AWS 액세스 포털 세션을 시작하는 설정이 포함되어 있습니다.

```
[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
```

AWS SDK for PHP 에서는 IAM Identity Center 인증을 사용하기 위해 애플리케이션에 추가 패키지(예: `SSO` 및 `SSOOIDC`)를 추가할 필요가 없습니다.

AWS 액세스 포털 세션 시작

에 액세스하는 애플리케이션을 실행하기 전에 SDK가 IAM Identity Center 인증을 사용하여 자격 증명을 확인하려면 활성 AWS 액세스 포털 세션이 AWS 서비스필요합니다. 구성된 세션 길이에 따라 결국 액세스가 만료되고 SDK에 인증 오류가 발생합니다. AWS 액세스 포털에 로그인하려면에서 다음 명령을 실행합니다 AWS CLI.

```
aws sso login
```

지침에 따라 기본 프로필을 설정했다면 `--profile` 옵션으로 명령을 직접적으로 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 `aws sso login --profile named-profile` 명령을 사용합니다.

활성 세션이 이미 있는지 선택적으로 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

세션이 활성 상태인 경우 이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고합니다.

> **Note**
>
> 이미 활성 AWS 액세스 포털 세션이 있고를 실행하는 경우 `aws sso login`자격 증명을 제공할 필요가 없습니다.
> 로그인 프로세스에서 데이터에 대한 AWS CLI 액세스를 허용하라는 메시지가 표시될 수 있습니다. AWS CLI 는 SDK for Python을 기반으로 구축되므로 권한 메시지에 `botocore` 이름의 변형이 포함될 수 있습니다.

인증에 대해 자세히 알아봅니다.

- IAM Identity Center를 인증에 사용하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 IAM Identity Center 인증 이해하기를 참조하세요.
- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 IAM의 보안 모범 사례를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 임시 보안 자격 증명을 참조하세요.
- AWS SDK for PHP 에서 사용할 수 있는 기타 자격 증명 공급자에 관해 알아보려면 AWS SDK 및 도구 참조 안내서에서 표준화된 보안 인증 공급자 섹션을 참조하세요.

AWS SDK for PHP 버전 3을 사용하여 간단한 애플리케이션 생성

AWS SDK for PHP를 사용하여 Amazon S3에 인사하세요. 다음 예제에서는 모든 Amazon S3 버킷의 목록을 표시합니다.

사전 조건

- SDK 다운로드 및 설치
- AWS SDK for PHP를 사용하기 전에 AWS 인증을 받아야 합니다. CHAP 인증 설정에 대한 자세한 내용은 AWS SDK for PHP 버전 3을 AWS 사용하여 로 인증를 참조하세요

코드에 SDK 포함

어떤 기술을 사용하여 SDK를 설치했든 상관없이, 단일 `require` 문을 사용하여 SDK를 코드에 포함할 수 있습니다. 설치 기법에 가장 적합한 PHP 코드를 찾으려면 다음 표를 참조하세요. `/path/to/`의 인스턴스를 시스템의 실제 경로로 바꿉니다.

설치 기법	Require 명령문
생성자 사용	`require '/path/to/vendor/autoload.php';`
phar 사용	`require '/path/to/aws.phar';`
ZIP 사용	`require '/path/to/aws-autoloader.php';`

이 항목에서는 Composer 설치 방법을 가정하는 예제를 보여 줍니다. 다른 설치 방법을 사용하는 경우 이 단원으로 돌아와서 사용할 올바른 `require` 코드를 찾을 수 있습니다.

코드 쓰기

인증이 가능한지 확인합니다.

다음 코드를 복사하여 새로운 소스 파일에 붙여 넣습니다. 이 파일을 저장하고 이름을 `hello-s3.php`로 지정합니다.

```
<?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";
}
```

프로그램 실행

명령 프롬프트를 PHP 프로그램을 실행합니다. PHP 프로그램을 실행하는 일반적인 명령 구문은 다음과 같습니다.

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

이 샘플 코드는 인수를 사용하지 않습니다. 이 코드를 실행하려면 명령 프롬프트에 다음을 입력합니다.

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

다음 단계

다른 많은 Amazon S3 작업을 테스트하려면 GitHub의 AWS 코드 예제 리포지토리를 확인하세요.

AWS SDK for PHP 버전 3과 AWS Cloud9 함께 사용

> **Note**
>
> 신규 고객은 더 이상 AWS Cloud9를 사용할 수 없습니다. AWS Cloud9의 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세히 알아보기

AWS Cloud9 은 클라우드에서 코드를 작성하고, 소프트웨어를 빌드, 실행, 테스트, 디버그, 릴리스하는 데 사용하는 다양한 도구가 들어 있는 웹 기반의 통합 개발 환경(IDE)입니다. AWS SDK for PHP에서 AWS Cloud9을 사용하여 브라우저에서 PHP 코드를 작성하고 실행할 수 있습니다. AWS Cloud9에는 코드 편집기와 터미널 등과 같은 도구가 포함되어 있습니다. AWS Cloud9 IDE는 클라우드 기반이므로

사무실, 집 또는 어디서나 인터넷에 연결된 컴퓨터를 사용하여 프로젝트 작업을 할 수 있습니다. AWS Cloud9에 대한 일반적인 내용은 AWS Cloud9 사용 설명서를 참조하세요.

다음 지침에 따라 AWS SDK for PHP에서 AWS Cloud9를 설정하세요.

- 1단계: AWS Cloud9을 사용하도록 AWS 계정 계정 설정
- 2단계: AWS Cloud9 개발 환경 설정
- 3단계: AWS SDK for PHP 설정
- 단계: 예제 코드 다운로드
- 단계: 예제 코드 실행

1단계: AWS Cloud9를 사용하도록 AWS 계정 설정

AWS Cloud9를 사용하려면 AWS Management Console에서 AWS Cloud9 콘솔에 로그인합니다.

> **Note**
>
> AWS IAM Identity Center을 사용하여 인증하는 경우 IAM 콘솔의 사용자 연결 정책에 필요한 `iam:ListInstanceProfilesForRole`의 권한을 추가해야 할 수 있습니다.

AWS 계정에 IAM 엔터티를 설정하여 AWS Cloud9에 액세스하고 AWS Cloud9 콘솔에 로그인하려면 AWS Cloud9 사용 설명서에서 AWS Cloud9의 팀 설정을 참조하세요.

2단계: AWS Cloud9 개발 환경 설정

AWS Cloud9 콘솔에 로그인한 후 콘솔을 사용하여 AWS Cloud9 개발 환경을 생성합니다. 환경을 생성하면 AWS Cloud9에서 해당 환경용 IDE를 엽니다.

자세한 내용은 AWS Cloud9 사용 설명서의 AWS Cloud9에서 환경 생성을 참조하세요.

> **Note**
>
> 콘솔에서 처음으로 환경을 생성할 때 Create a new instance for environment (EC2)(환경에 대한 새 인스턴스 생성(EC2)) 옵션을 선택하는 것이 좋습니다. 이 옵션은 AWS Cloud9에 환경을 생성하고 Amazon EC2 인스턴스를 시작한 다음, 새 인스턴스를 새 환경에 연결하도록 지시합니다. 이는 AWS Cloud9를 사용하는 가장 빠른 방법입니다.

터미널이 IDE에 아직 열려 있지 않은 경우 엽니다. IDE의 메뉴 모음에서 Window, New Terminal(창, 새 터미널)을 선택합니다. 터미널 창을 사용하여 도구를 설치하고 애플리케이션을 빌드할 수 있습니다.

3단계: 설정AWS SDK for PHP

AWS Cloud9에서 개발 환경용 IDE가 열리면 터미널 창을 사용하여 환경에서 AWS SDK for PHP를 설정합니다.

Composer를 통해 AWS SDK for PHP를 설치하는 것이 좋습니다. Composer는 프로젝트의 종속 항목을 관리 및 설치하는 PHP용 도구입니다.

Composer를 설치하고, 자동 로딩을 구성하고, 각종 모범 사례에 따라 종속 항목을 정의하는 방법에 대한 자세한 내용은 getcomposer.org를 참조하세요.

Composer 설치

프로젝트에 Composer가 없는 경우, Composer 다운로드 페이지에서 Composer를 다운로드 및 설치하세요.

- Windows의 경우, Windows 설치 프로그램 지침을 따르세요.
- Linux의 경우 명령줄 설치 지침을 따르세요.

Composer를 통해 종속 항목으로 AWS SDK for PHP 추가

시스템에 Composer가 이미 전역 설치되어 있다면 프로젝트의 기본 디렉터리에서 다음을 실행하여 AWS SDK for PHP를 종속 항목으로 설치합니다.

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

그렇지 않으면 이 Composer 명령을 입력하여 AWS SDK for PHP 최신 버전을 종속 항목으로 설치합니다.

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

php 스크립트에 자동 로더 추가

Composer 설치 작업은 사용자 환경에 여러 폴더와 파일을 만듭니다. 사용할 주 파일은 `autoload.php`이고 환경의 `vendor` 폴더에 있습니다.

스크립트에서 AWS SDK for PHP를 활용하려면 다음과 같이 스크립트에 자동 로더를 포함시켜야 합니다.

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

4단계: 예제 코드 다운로드

터미널 창을 사용하여 AWS SDK for PHP에 대한 예제 코드를 AWS Cloud9 개발 환경으로 다운로드합니다.

공식 AWS SDK 설명서에 사용되는 모든 코드 예의 복사본을 환경의 루트 디렉터리로 다운로드하려면 다음 명령을 실행하세요.

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

AWS SDK for PHP의 코드 예제는 `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` 디렉터리에 있습니다. 여기서 `ENVIRONMENT_NAME`은 개발 환경 이름입니다.

Amazon S3 예제를 사용하여 따라하려면 코드 `ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php` 예제로 시작하는 것이 좋습니다. 이 예제에서는 Amazon S3 버킷을 나열합니다. 터미널 창을 사용하여 `s3` 디렉터리로 이동하여 파일을 나열합니다.

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

AWS Cloud9에서 파일을 열려면 터미널 창에서 `ListBuckets.php`를 직접 클릭하면 됩니다.

코드 예제를 이해하는 데 도움이 더 필요하면 AWS SDK for PHP 코드 예제를 참조하세요.

5단계: 예제 코드 실행

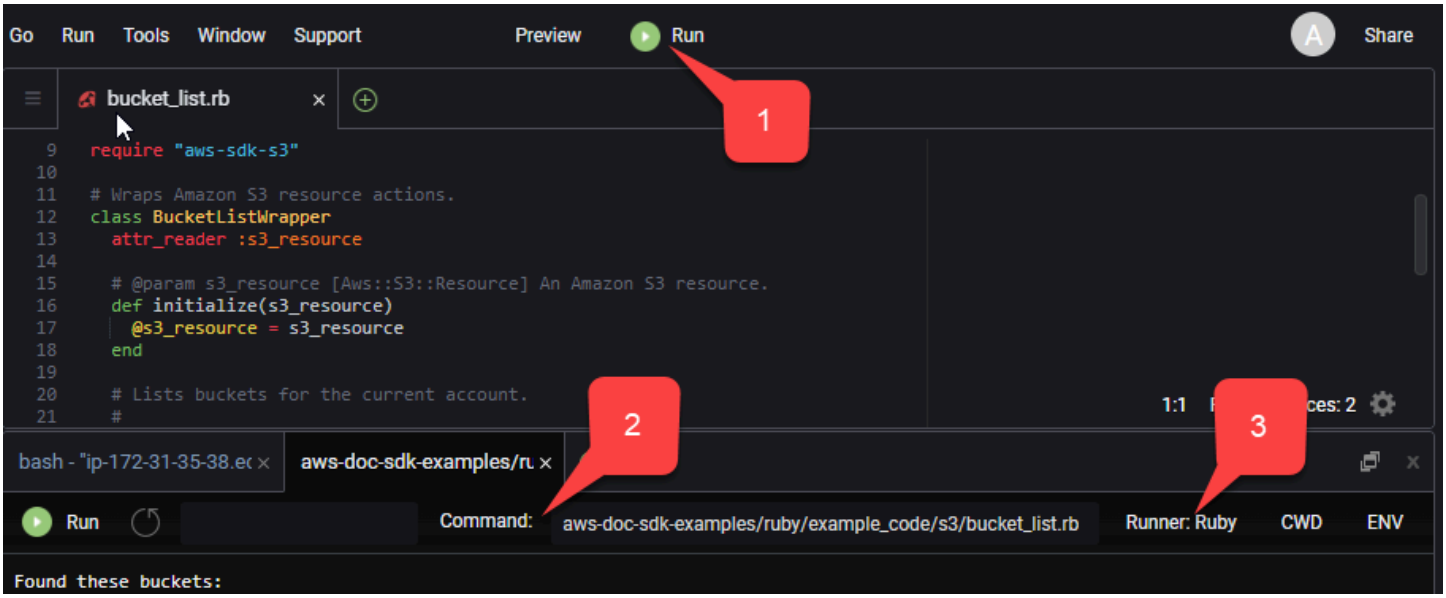
AWS Cloud9 개발 환경에서 코드를 실행하려면 상단 메뉴 표시줄에서 실행 버튼을 선택합니다. AWS Cloud9에서 `.php` 파일 확장자를 자동으로 감지하고 PHP(내장 웹 서버) 실행 프로그램을 사용하여 코드를 실행합니다. 하지만 이 예제에서는 실제로 PHP(**`cli`**) 옵션이 필요합니다. AWS Cloud9에서 코드 실행에 대한 자세한 정보는 AWS Cloud9 사용 설명서.의 코드 실행을 참조하세요.

다음 스크린샷에서 이러한 기본 영역을 확인하세요.

- 1: 실행. 실행 버튼은 상단 메뉴 표시줄에 있습니다. 이 버튼은 결과를 볼 수 있는 새 탭을 엽니다.

> **Note**
>
> 또한 새 실행 구성을 수동으로 생성할 수 있습니다. 메뉴 표시줄에서 실행, 실행 구성, 새 실행 구성을 선택합니다.

- 2: 명령. AWS Cloud9는 명령 텍스트 상자를 실행하는 파일의 경로와 파일 이름으로 채웁니다. 코드에서 명령줄 파라미터가 전달될 것으로 예상하는 경우 터미널 창을 통해 코드를 실행할 때와 동일한 방식으로 명령줄 파라미터를 명령줄에 추가할 수 있습니다.
- 3: 실행 프로그램. AWS Cloud9는 파일 확장자가 `.php`인지 감지하고 PHP(내장 웹 서버) 러너를 선택하여 코드를 실행합니다. 이 예제를 대신 실행하려면 PHP(**`cli`**) 를 선택하세요.



실행 코드에서 생성된 모든 출력이 탭에 표시됩니다.

AWS SDK for PHP 버전 3에서 서비스 클라이언트 구성

AWS 서비스에 프로그래밍 방식으로 액세스하기 위해 AWS SDK for PHP 버전 3에서는 각 AWS 서비스에 대해 클라이언트 객체를 사용합니다. 예를 들어, 애플리케이션에서 Amazon EC2에 액세스해야 하는 경우 해당 서비스와 인터페이스하기 위해 Amazon EC2 클라이언트 객체(Ec2Client 클래스의 인스턴스)를 생성합니다. 그런 다음 서비스 클라이언트를 사용하여 요청을 AWS 서비스에 보내면 됩니다.

SDK 동작을 구성하는 방법은 다양하지만, 궁극적으로 모든 것은 서비스 클라이언트의 동작과 관련이 있습니다. 모든 구성은 코드에서 해당 구성을 사용하는 서비스 클라이언트를 생성하기 전까지는 아무런 효과가 없습니다.

제공하는 구성의 예는 다음과 같습니다.

- 서비스 호출 시 코드로 AWS에 대해 인증하는 방식
- 서비스 클라이언트에서 사용할 AWS 리전
- 서비스 호출에 대한 재시도 및 제한 시간 설정
- HTTP 프록시 구성

여러 AWS SDK에 공통적으로 적용되는 설정, 기능 및 기타 기본 개념에 대해서는 AWS SDK 및 도구 참조 안내서를 참조하세요.

외부에서 AWS SDK for PHP 버전 3용 서비스 클라이언트 구성

코드 외부에서 많은 구성 설정을 처리할 수 있습니다. 대부분의 구성 설정은 환경 변수로 설정하거나 별도의 공유 AWS `config` 파일에서 설정할 수 있습니다. AWS 공유 `config` 파일은 프로필이라는 별도의 설정 세트를 유지하여 다양한 환경 또는 테스트에 대해 다양한 구성을 제공할 수 있습니다. AWS 공유 `config` 및 `credentials` 파일에 대한 자세한 설명은 AWS SDK 및 도구 참조 안내서를 참조하세요.

대부분의 환경 변수와 공유 `config` 파일 설정은 표준화되어 있으며 다양한 프로그래밍 언어와 애플리케이션에서 일관된 기능을 지원하기 위해 AWS SDK 및 도구 전반에 걸쳐 공유됩니다.

환경 변수 또는 구성 파일에서 SDK가 해결할 수 있는 모든 설정을 보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 설정 참조를 참조하세요.

클라이언트 구성을 위한 구성 공급자 체인

SDK는 여러 위치(또는 소스)를 확인하여 구성 값을 찾습니다.

1. 코드나 서비스 클라이언트 자체에 설정된 모든 명시적 설정은 다른 모든 설정보다 우선합니다.
2. 환경 변수
   - 환경 변수를 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에 나와 있는 환경 변수 섹션을 참조하세요.
   - 쉘 환경 변수는 시스템 전체, 사용자 전체, 특정 터미널 세션 등 다양한 수준에서 구성할 수 있습니다.
3. 공유 `config` 및 `credentials` 파일
   - 이 파일 설정에 관한 세부 사항을 알아보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 공유 `config` 및 `credentials` 파일 섹션을 참조하세요.
4. SDK 소스 코드 자체에서 제공하는 기본값이 마지막으로 사용됩니다.
   - 리전과 같은 일부 속성에는 기본값이 없습니다. 코드, 환경 설정 또는 공유 `config` 파일에서 명시적으로 지정해야 합니다. SDK가 필요한 구성을 해결할 수 없는 경우 API 요청이 런타임에 실패할 수 있습니다.

이 일반 구성 체인 외에도 AWS SDK for PHP 버전 3에는 자격 증명 공급자 체인 및 AWS 리전 해결 체인을 포함한 특수 공급자 체인도 사용됩니다. 이러한 특수 체인에는 SDK가 실행되는 환경을 고려하는 추가 공급자가 포함됩니다. 예를 들어 컨테이너나 EC2 인스턴스에서 실행되는 경우를 말합니다.

외부 설정을 사용하여 구성된 서비스 클라이언트 생성

AWS 서비스와 통신하려면 애플리케이션에 서비스 클라이언트를 생성해야 합니다. 서비스 클라이언트는 AWS 서비스와의 연결 수단으로, 복잡한 통신 세부 사항을 모두 처리하므로 사용자가 이를 걱정할 필요가 없습니다. 보안, 오류 처리 및 재시도와 같은 중요한 작업을 자동으로 처리하므로 기술적 복잡성을 처리하는 대신 애플리케이션 구축에 집중할 수 있습니다.

파라미터가 없는 생성자를 사용하여 서비스 클라이언트 구성

필요한 모든 구성 설정이 외부 소스에서 제공되는 경우 빈 생성자를 사용하여 서비스 클라이언트를 생성할 수 있습니다.

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

use Aws\S3\S3Client;
```

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

위의 코드 스니펫은 `S3Client` 인스턴스를 생성합니다. 생성 중에 SDK는 구성 공급자 체인에서 설정을 찾습니다. 설정 값을 찾으면 해당 값이 사용됩니다.

기본 AWS 리전 공급자 체인과 기본 자격 증명 공급자 체인도 생성 프로세스에 사용됩니다. 체인의 어느 지점에서 SDK는 사용할 AWS 리전을 확인하고 요청에 서명하기 위한 자격 증명을 검색할 수 있는 설정을 찾아야 합니다. 이러한 값을 찾지 못하면 클라이언트 생성이 실패합니다.

AWS SDK for PHP 버전 3 환경 변수

대부분의 AWS SDK에서 지원하는 cross-sdk 설정 외에도 AWS SDK for PHP 버전 3은 다음 환경 변수와 함께 작동합니다.

AWS_SDK_LOAD_NONDEFAULT_CONFIG

: 이 환경 변수를 설정하면 SDK는 AWS `config`파일(`~/.aws/config`) 외에도 `credentials` 파일(`~/.aws/credential`)에서 자격 증명을 로드합니다.

AWS_SDK_UA_APP_ID

: SDK에서 수행한 요청에 대해 User-Agent 헤더에 포함될 사용자 지정 애플리케이션 식별자를 설정합니다.

AWS_SUPPRESS_PHP_DEPRECATION_WARNING

: true로 설정하면 SDK에서 생성할 수 있는 PHP 사용 중단 경고가 표시되지 않습니다.

코드 내에서 AWS SDK for PHP 버전 3용 서비스 클라이언트 구성

서비스 클라이언트를 외부에서 구성하는 대신 또는 그에 추가하여 코드 내에서 프로그래밍 방식으로 구성할 수 있습니다.

코드 내에서 서비스 클라이언트를 구성하면 사용 가능한 다양한 옵션을 세밀하게 제어할 수 있습니다. 외부에서 설정할 수 있는 대부분의 구성도 코드 내에서 설정할 수 있습니다.

코드 내 기본 구성

클라이언트의 생성자에 결합형 배열의 옵션을 전달하여 코드에서 서비스 클라이언트를 생성하고 구성할 수 있습니다. 다음 예에서는 결합형 배열에 클라이언트가 사용하는 "region" 옵션만 포함되어 있습니다.

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

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

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

선택적 'version' 파라미터에 대한 정보는 구성 옵션 항목에서 확인할 수 있습니다.

보안 인증을 클라이언트에 명시적으로 제공하지 않았다는 점에 주의하세요. 이는 SDK가 기본 자격 증명 공급자 체인을 사용하여 자격 증명 정보를 찾기 때문입니다.

모든 일반 클라이언트 구성 옵션에 대해서는 클라이언트 생성자 옵션(AWS SDK for PHP 버전 3)에서 자세히 설명합니다. 클라이언트에 제공되는 옵션의 배열은 어떤 클라이언트를 생성하고 있는지에 따라 다를 수 있습니다. 이러한 사용자 지정 클라이언트 구성 옵션에 대해서는 각 클라이언트의 API 설명서에서 설명합니다.

Sdk 클래스 사용

Aws\Sdk 클래스는 클라이언트 팩토리 역할을 하며 여러 클라이언트 간의 공유 구성 옵션을 관리하는 데 사용됩니다. 특정 클라이언트 생성자에 제공할 수 있는 동일한 옵션을 Aws\Sdk 클래스에도 제공할 수 있습니다. 그러면 이러한 옵션은 각 클라이언트 생성자에 적용됩니다.

가져옵니다.

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

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

샘플 코드

```
// 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();
```

모든 클라이언트 간에 공유되는 옵션은 루트 수준 키-값 페어에 배치됩니다. 서비스별 구성 데이터는 서비스의 네임스페이스(예: "S3", "DynamoDb" 등)와 동일한 키를 갖는 결합형 배열에 제공할 수 있습니다.

```
$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();
```

서비스별 구성 값은 서비스별 값과 루트 수준 값의 결합입니다(즉, 서비스별 값은 루트 수준 값에 얕게 병합됨).

> **Note**
>
> 애플리케이션에서 여러 클라이언트 인스턴스를 사용하는 경우 Sdk 클래스를 사용하여 클라이언트를 생성하는 것이 좋습니다. Sdk 클래스는 각 SDK 클라이언트에 동일한 HTTP 클라이언트를 자동으로 사용하여 다른 서비스에 대한 SDK 클라이언트가 비차단 HTTP 요청을 수행할 수 있도록 허용합니다. SDK 클라이언트가 동일한 HTTP 클라이언트를 사용하지 않으면 SDK에서 전송된 HTTP 요청이 서비스 간의 promise 오케스트레이션을 차단할 수 있습니다.

클라이언트 생성자 옵션(AWS SDK for PHP 버전 3)

클라이언트 생성자를 클라이언트 생성자에 제공하거나 `Aws\Sdk` 클래스에 제공할 수 있습니다. 특정 유형의 클라이언트에 제공되는 옵션의 배열은 어떤 클라이언트를 생성하고 있는지에 따라 다를 수 있습니다. 이러한 사용자 지정 클라이언트 구성 옵션에 대해서는 각 클라이언트의 API 설명서에서 설명합니다.

클라이언트에 필요한 클라이언트 생성자 옵션을 명시적으로 제공하지 않으면 SDK for PHP는 환경 변수 또는 AWS 구성 파일에서 값을 찾습니다. 모든 클라이언트에는 자격 증명 공급자 값과 AWS 리전 값이 필요하므로 이러한 값을 생성자 옵션으로 설정하거나 외부에서 설정해야 합니다.

기본적으로 확인하는 구성 파일은 홈 디렉터리(일반적으로 `.aws/config`)에 있는 `~/.aws/config`입니다. 그러나 `AWS_CONFIG_FILE` 환경 변수를 사용하여 기본 구성 파일 위치를 설정할 수 있습니다. 예를 들어 이는 `open_basedir`가 있는 특정 디렉터리에 대한 파일 액세스를 제한하는 경우에 유용할 수 있습니다.

공유 AWS `config` 및 `credentials` 파일의 위치 및 형식에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 구성을 참조하세요.

AWS 구성 파일에서 또는 환경 변수로 설정할 수 있는 모든 글로벌 구성 설정에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 구성 및 인증 설정 참조를 참고하세요.

구성 옵션

- api_provider
- 자격 증명
- 디버그
- stats
- 엔드포인트
- endpoint_provider
- endpoint_discovery
- 핸들러
- http
- http_handler
- profile
- 리전
- retries
- scheme
- 서비스
- signature_provider
- signature_version

- ua_append
- use_aws_shared_config_files
- 검증
- version

다음 예제에서는 옵션을 Amazon S3 클라이언트 생성자에 전달하는 방법을 보여 줍니다.

```
use Aws\S3\S3Client;

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

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

클라이언트 만들기에 대한 자세한 내용은 the section called “기본 서비스 클라이언트 생성”을 참조하세요.

api_provider

유형

```
callable
```

유형, 서비스 및 버전 인수를 받아 해당 구성 데이터의 배열을 반환하는 PHP collable입니다. 유형 값은 `api`, `waiter`, `paginator` 중 하나일 수 있습니다.

기본적으로 SDK는 SDK의 `Aws\Api\FileSystemApiProvider` 폴더에서 API 파일을 로드하는 `src/data`의 인스턴스를 사용합니다.

자격 증명

유형

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

특정 보안 인증 인스턴스에 사용할 `Aws\Credentials\CredentialsInterface` 객체를 전달합니다. 다음은 IAM Identity Center 보안 인증 공급자를 사용해야 한다고 지정합니다. 이 공급자를 SSO 보안 인증 공급자라고도 합니다.

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

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

이름이 지정된 프로필을 사용하는 경우 이전 예제에서 프로필 이름을 `default`로 대체하세요. 이름이 지정된 프로필을 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 공유 `config` 및 `credentials` 파일을 참조하세요.

사용할 보안 인증 공급자를 지정하지 않고 보안 인증 공급자 체인을 사용하는 경우 인증 실패로 인한 오류 메시지는 일반적으로 일반적입니다. 보안 인증이 유효한지 확인하는 소스 목록의 마지막 공급자로부터 생성된 것으로, 사용하려는 공급자가 아닐 수도 있습니다. 사용할 보안 인증 공급자를 지정하면 표시되는 오류 메시지는 해당 공급자에서만 발생하므로 더 유용하고 관련성이 높습니다. 보안 인증을 확인하는 소스 체인에 대해 자세히 알아보려면 AWS SDK 및 도구 참조 가이드의 보안 인증 공급자 체인을 참조하세요.

null 보안 인증을 사용하고 요청에 서명하지 않으려면 `false`를 전달합니다.

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

함수를 사용하여 보안 인증을 생성하려면 collable 보안 인증 공급자 함수를 전달합니다.

```
use Aws\Credentials\CredentialProvider;

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

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

여러 프로세스 간의 기본 공급자 체인에서 반환된 값을 캐시하려면 `Aws\CacheInterface`의 인스턴스에 캐시된 보안 인증을 전달합니다.

```
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
]);
```

AWS SDK for PHP 버전 3의 보안 인증 가이드에서 보안 인증을 클라이언트에 제공하는 방법에 대한 자세한 내용을 참조할 수 있습니다.

> **Note**
>
> 보안 인증은 사용될 때 지연 로드되고 확인됩니다.

디버그

유형

```
bool|array
```

각 전송에 대한 디버그 정보를 출력합니다. 디버그 정보에는 트랜잭션이 준비되어 네트워크를 통해 전송될 때 트랜잭션의 각 상태 변경에 대한 정보가 포함됩니다. 클라이언트에 사용되는 특정 HTTP 핸들러에 대한 정보도 디버그 출력에 포함됩니다(예: cURL 출력 디버그).

요청을 정송할 때 디버그 정보를 표시하려면 `true`로 설정합니다.

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

```
]);

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

또한 다음 키와 함께 결합형 배열을 제공할 수 있습니다.

logfn (collable)

로그 메시지에서 호출되는 함수입니다. 기본적으로 PHP의 `echo` 함수가 사용됩니다.

stream_size (int)

스트림 크기가 이 숫자보다 크면 스트림 데이터가 로깅되지 않습니다. 스트림 데이터를 로깅하지 않으려면 `0`으로 설정합니다.

scrub_auth (bool)

로깅된 메시지에서 인증 데이터의 스크러빙을 비활성화하려면 `false`로 설정합니다(이렇게 하면 AWS 액세스 키 ID와 서명이 `logfn`에 전달됨).

http (bool)

하위 수준 HTTP 핸들러(예: 상세 cURL 출력)의 "디버그" 기능을 비활성화하려면 `false`로 설정합니다.

auth_headers (array)

매핑된 값을 교체할 값으로 교체하려고 하는 헤더의 키-값 매핑으로 설정합니다. 이러한 값은 `scrub_auth`를 `true`로 설정하지 않는 한 사용되지 않습니다.

auth_strings (array)

교체에 매핑할 정규식의 키-값 매핑으로 설정합니다. 이 값은 `scrub_auth`가 `true`로 설정된 경우 인증 데이터 스크러버에 사용됩니다.

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => [
        'logfn'        => function ($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**
>
> 이 옵션은 `http` 디버그 옵션에서 생성된 기본 HTTP 핸들러 정보도 출력합니다. 디버그 출력은 AWS SDK for PHP에서 문제를 진단할 때 매우 유용합니다. SDK에서 문제를 열 때 격리된 장애 사례에 대한 디버그 출력을 제공하세요.

stats

유형

```
bool|array
```

SDK 작업에서 반환된 오류 및 결과에 전송 통계를 바인딩합니다.

전송된 요청에 대한 전송 통계를 수집하려면 `true`로 설정합니다.

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

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

또한 다음 키와 함께 결합형 배열을 제공할 수 있습니다.

retries (bool)

시도된 재시도에 대한 보고를 비활성화하려면 `false`로 설정합니다. 기본적으로 재시도 통계가 수집되고 반환됩니다.

http (bool)

하위 수준 HTTP 어댑터에서 통계 수집을 활성화하려면 `true`로 설정합니다(예: GuzzleHttpTransferStats에서 반환된 값). 이 항목이 효과를 나타내려면 HTTP 핸들러가 __on_transfer_stats 옵션을 지원해야 합니다. HTTP 통계는 결합형 배열의 인덱싱된 배열로 반환되며, 각 결합형 배열에는 클라이언트의 HTTP 핸들러에서 요청에 대해 반환된 전송 통계가 포함됩니다. 기본 설정은 “Disable”입니다.

요청을 재시도한 경우 첫 번째 요청에 대한 통계가 포함된 `$result['@metadata']['transferStats']['http'][0]`, 두 번째 요청에 대한 통계가 포함된 `$result['@metadata']['transferStats']['http'][1]` 등으로 각 요청의 전송 통계가 반환됩니다.

timer (bool)

작업에 소비된 총 벽 시계(wall clock) 시간(초)을 보고하는 명령 타이머를 활성화하려면 `true`로 설정합니다. 기본 설정은 “Disable”입니다.

```
$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'];
```

엔드포인트

유형

```
string
```

웹 서비스의 전체 URI입니다. 이 항목은 AWS Elemental MediaConvert와 같이 계정별 엔드포인트를 사용하는 서비스에 필요합니다. 이러한 서비스의 경우 `describeEndpoints` 메서드를 사용하여 이 엔드포인트를 요청하세요.

이 항목은 사용자 지정 엔드포인트에 연결할 때만 필요합니다(예: Amazon S3 로컬 버전 또는 Amazon DynamoDB Local).

다음은 Amazon DynamoDB Local에 연결하는 예제입니다.

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

사용 가능한 AWS 리전과 엔드포인트의 목록은 AWS 리전 및 엔드포인트를 참조하세요.

endpoint_provider

유형

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

“서비스” 및 “리전” 키를 포함한 옵션의 해시를 수락하는 호출 가능한 EndpointProviderV2 또는 PHP(선택 사항)입니다. 이 항목은 NULL 또는 엔드포인트 데이터의 해시를 반환합니다. 여기서 "엔드포인트" 키는 필수입니다.

다음은 최소 엔드포인트 공급자를 생성하는 방법을 보여주는 예제입니다.

```
$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

유형

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

엔드포인트 검색은 엔드포인트 검색을 지원하는 서비스 API일 경우 정확한 엔드포인트를 찾아 연결합니다. 엔드포인트 검색을 지원하지만 필요하지 않은 서비스라면 클라이언트 생성 시 `endpoint_discovery`를 비활성화하세요. 서비스가 엔드포인트 검색을 지원하지 않는다면 이 구성은 무시됩니다.

`Aws\EndpointDiscovery\ConfigurationInterface`

서비스가 지정하는 작업에 적합한 서비스 API 엔드포인트에 자동으로 연결해주는 구성 공급자(선택 사항)입니다.

`Aws\EndpointDiscovery\Configuration` 객체는 엔드포인트 검색의 활성화 여부를 나타내는 부울 값("enabled")과 엔드포인트 캐시의 최대 키 수를 나타내는 정수("cache_limit")를 포함해 두 가지 옵션을 허용합니다.

클라이언트가 생성될 때마다 `Aws\EndpointDiscovery\Configuration` 객체를 전달하여 엔드포인트 검색을 위한 특정 구성을 사용하세요.

```
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,
```

```
]);
```

다수의 프로세스에서 엔드포인트 검색으로 반환되는 값을 캐싱하려면 Aws\CacheInterface 인스턴스를 전달합니다.

```
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),
]);
```

배열을 엔드포인트 검색에 전달합니다.

```
use Aws\S3\S3Client;

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

핸들러

유형

callable

명령 객체 및 요청 객체를 받아 GuzzleHttp\Promise\PromiseInterface 객체와 함께 실행되거나 Aws\ResultInterface과 함께 거부되는 promise(Aws\Exception\AwsException)를 반환하는 핸들러입니다. 핸들러는 다음 핸들러를 받지 않습니다. 왜냐하면 다음 핸들러는 터미널이며 명령을 이행할 것이기 때문입니다. 핸들러를 제공하지 않으면 기본 Guzzle 핸들러가 사용됩니다.

Aws\MockHandler를 사용하여 모의(mock) 결과를 반환하거나 모의(mock) 예외를 발생할 수 있습니다. 결과 또는 예외를 대기열에 넣으면 MockHandler가 FIFO 순서로 결과 또는 예외를 대기열에서 제거합니다.

```
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

유형

```
array
```

SDK에서 생성된 HTTP 요청 및 전송에 적용되는 HTTP 옵션의 배열로 설정합니다.

SDK는 다음과 같은 구성 옵션을 지원합니다.

cert

유형

```
string|array
```

PEM 형식의 클라이언트 측 인증서를 지정합니다.

- 인증서 파일에 대한 경로를 문자열로 설정합니다.

```
use Aws\S3\S3Client;

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

- 경로 및 암호를 포함한 배열로 설정합니다.

```
use Aws\S3\S3Client;

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

connect_timeout

서버에 연결하려고 시도하는 동안 대기할 시간(초)을 설명하는 부동 소수점입니다. 무한정으로 대기하려면 `0`을 사용합니다(기본 동작).

```
use Aws\DynamoDb\DynamoDbClient;

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

```
    ]
]);
```

디버그

유형

```
bool|resource
```

기본 HTTP 핸들러에 디버그 정보를 출력하도록 지시합니다. 다른 HTTP 핸들러에서 제공되는 디버그 정보는 다를 수 있습니다.

- 디버그 출력을 STDOUT에 쓰려면 `true`를 전달합니다.
- 디버그 출력을 특정 PHP 스트림 리소스에 쓰려면 `resource`에서 반환되는 `fopen`를 전달합니다.

decode_content

유형

```
bool
```

기본 HTTP 핸들러에 압축된 응답의 본문을 inflate하도록 지시합니다. 이 항목을 활성화하지 않으면 압축된 응답 본문이 `GuzzleHttp\Psr7\InflateStream`으로 inflate될 수 있습니다.

> **Note**
>
> SDK의 기본 HTTP 핸들러에서 콘텐츠 디코딩이 기본적으로 활성화됩니다. 이전 버전과 호환성을 유지하기 위해 이 기본값을 변경할 수 없습니다. Amazon S3에서 압축된 파일을 저장하는 경우 S3 클라이언트 수준에서 콘텐츠 디코딩을 비활성화하는 것이 좋습니다.
>
> ```
> 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

유형

```
int
```

요청을 전송하기 전에 지연할 시간(밀리초)입니다. 이 항목은 요청을 다시 시도하기 전에 시간을 지연하기 위해 주로 사용됩니다.

expect

유형

```
bool|string
```

이 옵션은 기본 HTTP 핸들러로 전달됩니다. 기본적으로 요청 본문이 1MB를 초과하면 Expect: 100-계속 헤더가 설정됩니다. `true` 또는 `false`는 모든 요청에서 이 헤더를 각각 활성화하거나 비활성화합니다. 정수가 사용되는 경우에는 이 설정을 초과하는 본문의 요청만 헤더를 사용합니다. 정수로 사용될 때 Expect 헤더가 전송되는 본문 크기는 알 수 없습니다.

> **Warning**
>
> Expect 헤더를 비활성화하면 서비스가 인증 또는 기타 오류를 반환하지 못합니다. 따라서 이 옵션은 신중하게 구성해야 합니다.

progress

유형

```
callable
```

전송을 진행할 때 호출할 함수를 정의합니다. 이 함수에 사용할 수 있는 인수는 다음과 같습니다.

1. 다운로드할 총 예상 바이트 수입니다.
2. 지금까지 다운로드된 바이트 수입니다.
3. 업로드할 예상 바이트 수입니다.
4. 지금까지 업로드된 바이트 수입니다.

```
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

유형

```
string|array
```

proxy 옵션을 사용하여 프록시를 통해 AWS 서비스에 연결할 수 있습니다.

- 프록시에 연결하기 위한 문자열 값을 모든 유형의 URI에 제공합니다. 프록시 문자열 값에는 체계, 사용자 이름 및 암호가 포함될 수 있습니다. 예를 들어 "`http://username:password@192.168.16.1:10`"입니다.
- 키가 URI의 체계이고 값이 제공된 URI에 대한 프록시인 프록시 설정의 결합형 배열을 제공합니다(즉, “http” 및 “https” 엔드포인트에 다른 프록시를 제공할 수 있음).

```
use Aws\DynamoDb\DynamoDbClient;

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

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

HTTP_PROXY 환경 변수를 사용하여 “http” 프로토콜에 특정한 프록시를 구성할 수 있으며 HTTPS_PROXY 환경 변수를 사용하여 “https”에 특정한 프록시를 구성할 수 있습니다.

sink

유형

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

`sink` 옵션은 작업의 응답 데이터가 다운로드되는 위치를 제어합니다.

- 응답 본문을 PHP 스트림에 다운로드하려면 `resource`에서 반환되는 `fopen`를 제공합니다.

- 응답 본문을 디스크의 특정 파일에 다운로드하려면 디스크의 파일에 대한 경로를 `string` 값으로 제공합니다.
- 응답 본문을 특정 PSR 스트림 객체에 다운로드하려면 `Psr\Http\Message\StreamInterface`을 제공합니다.

> **Note**
>
> 기본적으로 SDK는 응답 본문을 PHP 임시 스트림에 다운로드합니다. 본문 크기가 2MB에 도달할 때까지 데이터가 메모리에 유지되며, 이 크기부터는 데이터가 디스크의 임시 파일에 기록됩니다.

synchronous

유형

`bool`

`synchronous` 옵션은 사용자가 결과를 차단하려고 한다고 기본 HTTP 핸들러에 알립니다.

스트림

유형

`bool`

모두 사전에 다운로드하는 대신 웹 서비스에서 응답의 응답 본문을 스트리밍하려고 한다고 기본 HTTP 핸들러에 알리려면 `true`로 설정합니다. 예를 들어, Amazon S3 스트림 래퍼 클래스에서 이 옵션을 이용하여 데이터가 스트리밍되도록 보장합니다.

제한 시간

유형

`float`

요청의 제한 시간(초)을 설명하는 부동 소수점입니다. 무한정으로 대기하려면 `0`을 사용합니다(기본 동작).

```
use Aws\DynamoDb\DynamoDbClient;

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

verify

유형

```
bool|string
```

`verify` `http` 옵션을 사용하여 SDK의 피어 SSL/TLS 인증서 확인 동작을 사용자 지정할 수 있습니다.

- SSL/TLS 피어 인증서 확인을 활성화하고 운영 체제에서 제공된 기본 CA 번들을 사용하려면 `true`로 설정합니다.
- 피어 인증서 확인을 비활성화하려면 `false`로 설정합니다. (이 설정은 안전하지 않음)
- 사용자 지정 CA 번들을 사용하여 확인을 활성화하기 위해 CA 인증서 번들에 대한 경로를 제공하려면 문자열로 설정합니다.

시스템에 대한 CA 번들을 찾을 수 없고 오류를 수신하는 경우 CA 번들에 대한 경로를 SDK에 제공합니다. 특정 CA 번들이 필요 없는 경우 Mozilla는 여기에서 다운로드할 수 있는 일반적으로 사용되는 CA 번들을 제공합니다(이 번들은 cURL 유지 관리자가 유지 관리함). 디스크에서 사용 가능한 CA 번들이 있으면 파일에 대한 경로를 가리키도록 `openssl.cafile` PHP .ini 설정을 설정하여 `verify` 요청 옵션을 생략할 수 있습니다. cURL 웹 사이트에서 SSL 인증서에 대한 자세한 내용을 참조할 수 있습니다.

```
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

유형

```
callable
```

`http_handler` 옵션은 SDK를 다른 HTTP 클라이언트와 통합하기 위해 사용됩니다. `http_handler` 옵션은 명령에 적용되는 `Psr\Http\Message\RequestInterface` 객체 및 `http` 옵션 배열을 받아 `GuzzleHttp\Promise\PromiseInterface` 객체를 통해 이행되거나 다음 예외 데이터의 배열을 통해 거부되는 `Psr\Http\Message\ResponseInterface` 객체를 반환하는 함수입니다.

- `exception` - (`\Exception`) 발생한 예외입니다.
- `response` - (`Psr\Http\Message\ResponseInterface`) 수신된 응답입니다(있는 경우).
- `connection_error` - (bool) 오류를 연결 오류로 표시하려면 `true`로 설정합니다. 이 값을 `true`로 설정하면 필요한 경우 SDK에서 작업을 자동으로 재시도할 수 있습니다.

SDK는 제공된 `http_handler`를 `handler` 객체로 래핑하여 제공된 `http_handler`를 일반 `Aws\WrappedHttpHandler` 옵션으로 자동으로 변환합니다.

기본적으로 SDK는 Guzzle을 HTTP 핸들러로 사용합니다. 여기에 다른 HTTP 핸들러를 제공하거나 Guzzle 클라이언트에 자체 사용자 지정 옵션을 제공할 수 있습니다.

TLS 버전 설정

한 가지 사용 사례는 Curl이 사용자 환경에 설치되어 있다고 가정할 때 Curl과 함께 Guzzle에서 사용하는 TLS 버전을 설정하는 것입니다. 지원되는 TLS 버전에 대한 Curl 버전 제약 조건에 유의하세요. 기본적으로 최신 버전이 사용됩니다. TLS 버전이 명시적으로 설정되어 있고 원격 서버가 이 버전을 지원하지 않으면 이전 TLS 버전이 사용되지 않고 오류가 발생합니다.

지정된 클라이언트 작업에 사용되고 있는 TLS 버전은 debug 클라이언트 옵션을 true로 설정하고 SSL 연결 출력을 검사하여 확인할 수 있습니다. 해당 행은 다음과 같을 것입니다. `SSL connection using TLSv1.2`

Guzzle 6으로 TLS 1.2를 설정하는 예제:

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

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

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

> **Note**
>
> `http_handler` 옵션은 제공된 모든 `handler` 옵션을 대체합니다.

profile

유형

`string`

"profile" 옵션은 홈 디렉터리의 보안 인증 파일 (일반적으로 `~/.aws/credentials`)에서 AWS 보안 인증을 생성할 때 사용할 프로필을 지정합니다. 이 설정은 `AWS_PROFILE` 환경 변수를 재정의합니다.

> **Note**
>
> "프로필" 옵션을 지정하면 "credentials" 옵션이 무시되고 AWS 구성 파일의 보안 인증 관련 (일반적으로 ~/.aws/config)은 무시됩니다.

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

AWS SDK for PHP 버전 3의 보안 인증 및 .ini 파일 형식 구성에 대한 자세한 내용은 버전 3의 보안 인증을 참조하세요.

리전

유형

`string`

필수

true

연결할 AWS 리전입니다. 사용 가능한 리전의 목록은 AWS 리전 및 엔드포인트를 참조하세요.

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

retries

유형

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

Default

```
int(3)
```

클라이언트에 대한 재시도 모드와 허용된 최대 재시도 횟수를 구성합니다. 재시도를 비활성화하려면 `0`을 전달합니다.

세 가지 재시도 모드는 다음과 같습니다.

- `legacy`- 기본 레거시 재시도 구현
- `standard`- 성공할 것 같지 않은 재시도를 방지하기 위해 재시도 할당량 시스템을 추가합니다
- `adaptive` - 표준 모드를 기반으로 빌드하여 클라이언트 측 속도 제한기 추가. 이 모드는 실험적인 것으로 간주됩니다.

재시도 구성은 각 요청에 사용할 모드와 최대 시도 횟수로 구성됩니다. 구성은 다음과 같은 우선 순위에 따라 서로 다른 두 위치에서 설정할 수 있습니다.

우선순위

재시도 구성의 우선 순위는 다음과 같습니다(1이 2-3보다 우선 적용).

1. 클라이언트 구성 옵션
2. 환경 변수
3. AWS공유 구성 파일

환경 변수:

- `AWS_RETRY_MODE` - `legacy`, `standard` 또는 `adaptive`로 설정
- `AWS_MAX_ATTEMPTS` - 요청당 최대 시도 횟수에 대해 정수 값으로 설정

공유 구성 파일 키

- `retry_mode` - `legacy`, `standard` 또는 `adaptive`로 설정
- `max_attempts` - 요청당 최대 시도 횟수에 대해 정수 값으로 설정

클라이언트 구성

다음 예제에서는 Amazon DynamoDB 클라이언트에 대한 재시도를 비활성화합니다.

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

다음 예제에서는 정수를 전달합니다. 기본값은 전달된 재시도 횟수가 있는 `legacy` 모드입니다.

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

`Aws\Retry\Configuration` 객체에는 두 개의 파라미터, 즉 재시도 모드와

요청당 최대 시도 횟수에 대한 정수를 사용할 수 있습니다. 이 예제에서는 재시도 구성에 대해

`Aws\Retry\Configuration` 객체를 전달합니다.

```
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,
]);
```

이 예제에서는 재시도 구성에 대해 배열을 전달합니다.

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

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

이 예제에서는 `Aws\CacheInterface`의 인스턴스를 전달하여 기본 재시도 구성 공급자가 반환한 값을 캐시합니다.

```
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

유형

`string`

Default

`string(5) "https"`

연결할 때 사용할 URI 체계입니다. 기본적으로 SDK는 “https” 엔드포인트를 사용합니다(즉, SSL/TLS 연결 사용). `scheme`을 “http”로 설정하여 암호화되지 않은 “http” 엔드포인트를 통해 서비스에 연결하려고 시도할 수 있습니다.

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

엔드포인트 목록과 서비스가 해당 `http` 체계를 지원하는지 여부는 AWS 지역 및 엔드포인트를 참조하세요.

서비스

유형

`string`

필수

true

사용할 서비스의 이름입니다. 이 값은 SDK에서 제공되는 클라이언트(즉, `Aws\S3\S3Client`)를 사용하면 기본적으로 제공됩니다. 이 옵션은 SDK에서 아직 게시되지 않았지만 디스크에서 사용할 수 있는 서비스를 테스트할 때 유용합니다.

signature_provider

유형

`callable`

서명 버전 이름(예: v4), 서비스 이름 및 AWS 리전을 받아 `Aws\Signature\SignatureInterface` 객체를 반환하거나 공급자가 지정된 파라미터에 대한 서명자를 생성할 수 있는 경우 `NULL` 수집 가능입니다. 이 공급자는 클라이언트에서 사용되는 서명자를 생성하는 데 사용됩니다.

`Aws\Signature\SignatureProvider` 클래스에는 SDK에서 제공되는 다양한 함수가 있으며 이러한 함수를 사용하여 사용자 지정 서명 공급자를 생성할 수 있습니다.

signature_version

유형

`string`

서비스에 사용할 사용자 지정 서명 버전을 나타내는 문자열입니다(예: v4). 작업에 따라 필요한 경우 서명 버전이 이 요청된 서명 버전을 재정의할 수 있습니다.

다음 예는 서명 버전 4를 사용하도록 Amazon S3 클라이언트를 구성하는 방법을 보여줍니다.

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

```
    'region'            => 'us-west-2',
    'signature_version' => 'v4'
]);
```

> **Note**
>
> 클라이언트에서 사용되는 signature_provider는 사용자가 제공하는 signature_version 옵션을 생성할 수 있어야 합니다. SDK에서 사용되는 기본 signature_provider는 “v4” 및 “익명” 서명 버전에 대한 서명 객체를 생성할 수 있습니다.

ua_append

유형

```
string|string[]
```

Default

```
[]
```

HTTP 핸들러에 전달된 사용자-에이전트 문자열에 추가되는 문자열 또는 문자열의 배열입니다.

use_aws_shared_config_files

유형

```
bool|array
```

Default

```
bool(true)
```

'~/.aws/config' 및 '~/.aws/credentials'에서 공유 구성 파일 검사를 비활성화하려면 거짓으로 설정합니다. AWS_CONFIG_FILE 환경 변수를 재정의합니다.

검증

유형

```
bool|array
```

Default

bool(true)

클라이언트 측 파라미터 확인을 비활성화하려면 false로 설정합니다. 확인을 끄면 클라이언트 성능이 약간 향상되지만 차이는 무시할 수 있는 정도입니다.

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

특정 확인 제약 조건을 활성화하려면 확인 옵션의 결합형 배열로 설정합니다.

- required - 필수 파라미터가 있는지 확인합니다(기본적으로 켜짐).
- min - 값의 최소 길이를 확인합니다(기본적으로 켜짐).
- max - 값의 최대 길이를 확인합니다.
- pattern - 값이 정규식과 일치하는지 확인합니다.

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

version

유형

string

필수

false

사용할 웹 서비스의 버전입니다(예: 2006-03-01).

SDK 버전 3.277.10부터는 "버전” 옵션이 필요하지 않습니다. ‘version’ 옵션을 지정하지 않으면 SDK에서는 최신 버전의 서비스 클라이언트를 사용합니다.

서비스 클라이언트를 만들 때 'version' 파라미터가 필요한 두 가지 상황이 있습니다.

- 3.277.10 이전 버전의 PHP SDK를 사용합니다.
- 버전 3.277.10 이상을 사용하고 있는데 서비스 클라이언트에 '최신' 버전이 아닌 다른 버전을 사용하려고 합니다.

예를 들어 다음 스니펫은 SDK 버전 3.279.7을 사용하지만 `Ec2Client`의 최신 버전은 사용하지 않습니다.

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

버전 제약 조건을 지정하면 서비스가 갑작스럽게 변경되더라도 코드에 영향을 미치지 않습니다.

사용 가능한 API 버전 목록은 각 클라이언트의 API 설명서 페이지에서 확인할 수 있습니다. 특정 API 버전을 로드할 수 없는 경우 설치된 SDK를 업데이트해야 할 수 있습니다.

AWS SDK for PHP 버전 3의 AWS 리전 설정

SDK 클라이언트는 클라이언트를 생성할 때 지정한 특정 AWS 리전에 있는 AWS 서비스에 연결됩니다. 이 구성을 사용하면 애플리케이션이 해당 지역의 AWS 리소스와 상호 작용할 수 있습니다. 리전을 명시적으로 설정하지 않고 서비스 클라이언트를 생성하면 외부 구성의 기본 리전이 사용됩니다.

리전 화인 체인

AWS SDK for PHP 버전 3에서는 다음 순서를 사용하여 서비스 클라이언트에서 사용할 리전을 결정합니다.

1. 코드에 제공된 리전 - 클라이언트 생성자 옵션에 리전을 명시적으로 설정한 경우 이 설정이 다른 모든 소스보다 우선합니다.

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

```
]);
```

2. 환경 변수 - 코드에 리전을 제공하지 않은 경우 SDK는 다음 환경 변수를 순서대로 확인합니다.
   - AWS_REGION
   - AWS_DEFAULT_REGION

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

3. AWS config 파일 - 리전 환경 변수가 설정되지 않은 경우 SDK는 AWS config 파일을 확인합니다.
   a. ~/.aws/config(또는 AWS_CONFIG_FILE 환경 변수로 지정된 위치)를 찾습니다.
   b. AWS_PROFILE 환경 변수로 지정된 프로필 내의 리전 설정을 확인합니다.
   c. AWS_PROFILE이 지정되지 않은 경우 "기본" 프로필을 사용합니다.

   예를 들어 다음과 같은 구성 파일 설정이 있다고 가정해 보겠습니다.

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

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

   AWS_PROFILE 환경 변수가 "production" 값으로 설정된 경우 클라이언트에서는 eu-central-1 Region을 사용합니다. AWS_PROFILE 환경 변수가 없는 경우 클라이언트에서는 eu-west-1 리전을 사용합니다.
4. 리전 값은 서비스 클라이언트의 필수 설정이므로 SDK가 위의 소스에서 리전 값을 찾지 못하면 예외가 발생합니다.

모범 사례

AWS SDK for PHP 버전 3에서 리전으로 작업할 때는 다음 모범 사례를 고려하세요.

프로덕션 코드에 명시적으로 리전 설정

프로덕션 애플리케이션의 경우 환경 변수 또는 config에 의존하지 않고 코드에 리전을 명시적으로 설정하는 것이 좋습니다. 이렇게 하면 코드의 예측 가능성이 높아지고 외부 구성에 대한 의존성이 줄어듭니다.

개발 및 테스트에 환경 변수 사용

개발 및 테스트 환경의 경우 환경 변수를 사용하면 코드를 변경하지 않고도 더 많은 유연성을 확보할 수 있습니다.

여러 환경에 프로필 사용

애플리케이션이 여러 AWS 환경에서 작동해야 하는 경우 AWS `config` 파일에서 여러 프로필을 사용하고 필요에 따라 이들 간에 전환하는 것이 좋습니다.

AWS SDK for PHP 버전 3 자격 증명 공급자 사용

AWS SDK에 사용할 수 있는 자격 증명 메커니즘에 대한 참조 정보는 AWS SDK 및 도구 참조 안내서에서 자격 증명 및 액세스를 참조하세요.

> **Important**
>
> 보안을 위해 루트 계정을 사용하여 AWS 액세스하지 않는 것이 좋습니다. 최신 보안 권장 사항은 항상 IAM 사용자 설명서의 IAM의 보안 모범 사례를 참조하세요.

AWS SDK for PHP 버전 3에서 자격 증명 공급자의 역할은 자격 증명을 소싱하고 SDK의 AWS 서비스 클라이언트에 제공하는 것입니다. SDK는 소싱한 자격 증명을 사용하여 각 요청에 암호화 방식으로 서명하여 서비스를 인증합니다. 자격 증명은 일반적으로 액세스 키(액세스 키 ID)와 시크릿 액세스 키로 구성됩니다.

IAM Identity Center 인증을 설정하거나 런타임이 IAM 역할을 수임하도록 구성하는 경우와 같이 임시 자격 증명을 사용하는 경우, 액세스 키에 세션 토큰이 추가되어 AWS에 대한 시간 제한이 있는 액세스를 제공합니다.

AWS SDK for PHP 버전 3에서 자격 증명 공급자란 무엇인가요?

보안 인증 공급자는 `GuzzleHttp\Promise\PromiseInterface` 인스턴스를 통해 이행되거나 `Aws\Credentials\CredentialsInterface`을 통해 거부되는 `Aws\Exception\CredentialsException`를 반환하는 함수입니다. SDK는 자격 증명 공급자 함수의 여러 구현을 제공하며, 자격 증명을 생성하거나 자격 증명 로딩을 최적화하기 위해 고유한 사용자 지정 로직을 구현할 수도 있습니다.

보안 인증 공급자는 `credentials` 클라이언트 생성자 옵션에 전달됩니다. 보안 인증 공급자는 비동기적이므로, API 작업을 호출할 때마다 강제로 늦게 평가됩니다. 따라서 보안 인증 공급자 함수를 SDK

클라이언트 생성자에 전달하면 보안 인증이 즉시 확인되지 않습니다. 보안 인증 공급자가 보안 인증 객체를 반환하지 않는 경우 API 작업은 `Aws\Exception\CredentialsException`을 통해 거부됩니다.

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

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

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

AWS SDK for PHP 버전 3의 기본 자격 증명 공급자 체인 이해

기본 자격 증명 공급자 체인은 SDK가 호출하는 일련의 기본 제공 자격 증명 공급자로 구성됩니다. 구현할 때는 파라미터 없이 defaultProvider 자격 증명 공급자 함수를 사용합니다. 유효한 보안 인증 정보를 찾은 후에는 검색이 중지됩니다.

는 다음 순서로 자격 증명 공급자를 AWS SDK for PHP 실행합니다.

- **env 공급자** - 환경 변수로 설정된AWS 액세스 키를 검색합니다.
- **assumeRoleWithWebIdentityCredentialProvider 공급자** - IAM 역할 및 웹 ID 토큰 파일 설정을 검색합니다.
- 체인의이 시점에서 SDK는 공유 AWS `config` 및 `credentials` 파일에서 구성을 찾습니다. SDK에서는 "기본" 프로필에서 구성을 검색하지만 `AWS_PROFILE` 환경 변수가 설정된 경우에는 명명된 프로필 값을 사용합니다.
  - **sso 공급자** - 공유 `config` 파일에서 IAM Identity Center 구성 설정을 찾습니다.
  - **login provider** - SDK는 공유 `config` 파일에서 AWS 콘솔 로그인 세션 구성 설정을 찾습니다.
  - **process 공급자** - 공유 `credentials` 파일에서 `credential_process` 설정을 찾습니다.
  - **ini 공급자** - SDK는 공유 `credentials` 파일에서 AWS 자격 증명 또는 IAM 역할 정보를 찾습니다.

  - **process 공급자** - 공유 config 파일에서 credential_process 설정을 찾습니다.
  - **ini 공급자** - SDK는 공유 config 파일에서 AWS 자격 증명 또는 IAM 역할 정보를 찾습니다.

- **ecsCredentials 공급자** - 임시 자격 증명을 얻는 데 필요한 정보를 제공하는 환경 변수 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 또는 AWS_CONTAINER_CREDENTIALS_FULL_URI를 찾습니다.
- **instanceProfile 공급자** - EC2 인스턴스 메타데이터 서비스를 사용하여 인스턴스 프로파일에 지정된 IAM 역할을 가져옵니다. 역할 정보를 사용하여 임시 자격 증명을 얻습니다.

> **Note**
>
> 기본 공급자의 결과는 자동으로 메모이제이션(memoization)됩니다.

체인에 대한 코드는 GitHub 소스 코드에서 확인할 수 있습니다.

AWS SDK for PHP 버전 3의 기본 제공 자격 증명 공급자

SDK는 여러 기본 제공 자격 증명 공급자를 제공하며, 이를 개별적으로 사용하거나 사용자 지정 자격 증명 공급자 체인으로 결합할 수 있습니다.

서비스 클라이언트 생성 중에 자격 증명 공급자를 지정하면 SDK는 지정된 자격 증명 공급자만 사용하여 자격 증명을 로드하려고 시도합니다. 기본 자격 증명 공급자 체인은 사용하지 않습니다. 서비스 클라이언트가 instanceProfile 공급자를 사용하도록 하려는 경우 서비스 클라이언트 생성자에 instanceProfile 공급자를 지정하여 기본 체인을 단축(short-circuit)시킬 수 있습니다.

```
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**
>
> API 작업을 수행할 때마다 보안 인증 공급자가 호출됩니다. 보안 인증 로딩이 비용이 많이 드는 작업(예: 디스크 또는 네트워크 리소스에서 로딩)이거나 공급자가 보안 인증일 캐시하지 않는 경우 보안 인증 공급자를 `Aws\Credentials\CredentialProvider::memoize` 함수 안에 래핑하는 것을 고려합니다. SDK에서 사용되는 기본 보안 인증 공급자는 자동으로 메모이제이션(memoization)됩니다.

주제

SDK for PHP의 **`login`** 공급자

`Aws\Credentials\CredentialProvider::login`는 AWS CLI와 같은 도구에서 지원하는 브라우저 기반 로그인 세션에서 구성된 자격 증명을 로드하려고 시도합니다. 인증 후 AWS SDKs 및 도구에서 작동하는 임시 자격 증명을 AWS 생성합니다.

이 프로세스를 사용하면 초기 계정 설정 중에 생성된 루트 자격 증명, IAM 사용자 또는 자격 증명 공급자의 페더레이션 자격 증명을 사용하여 인증할 수 있으며, PHP용 AWS SDK가 자동으로 임시 자격 증명을 관리합니다. 이 접근 방식은 장기 자격 증명을 로컬에 저장할 필요가 없으므로 보안을 강화합니다.

`aws login` 명령을 실행하면 활성 콘솔 세션에서 선택하거나 브라우저 기반 인증 흐름을 통해 로그인할 수 있으며, 그러면 임시 자격 증명이 자동으로 생성됩니다. PHP용 AWS SDK는 로그인 서비스를 사용하여 최대 12시간 동안 이러한 자격 증명을 자동으로 새로 고칩니다.

로그인 공급자는 제공된 프로필을 기반으로 이전에 언급한 로그인 세션 워크플로에서 생성된 액세스 토큰을 로드하려고 시도합니다. 공급자를 호출할 때 프로필이 제공되지 않으면 프로필로 돌아가기 전에 먼저 `AWS_PROFILE` 환경 변수를 확인하여 프로필을 확인하려고 시도합니다`default`. 인코드 구성은 공급자에게 전달될 수 있으며, 공급자가 자격 증명을 새로 고치는 데 사용되는 로그인 서비스 클라이언트의 `region` 값을 찾습니다. 구성 배열에 리전이 제공되지 않은 경우 공급자는 `AWS_REGION` 환경 변수를 확인한 다음 확인된 프로파일에 설정된 리전 값을 확인하여 리전을 확인하려고 시도합니다. 리전을 찾을 수 없는 경우 공급자는 리전을 구성하는 방법에 대한 지침과 함께 거부된 promise를 반환합니다.

공급자는 기본 체인의 일부로 호출되며 직접 호출할 수 있습니다.

```
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
]);
```

기본적으로 사용하려는 서비스 클라이언트에 자격 증명 구성이 제공되지 않은 경우이 공급자는 자격 `defaultProvider()` 증명 체인의 일부로 호출됩니다. 이 시나리오에서는 서비스 클라이언트의 리전이 자동으로 `login()` 공급자에게 전달됩니다. 또한이 시나리오에서는 프로필로 돌아가기 전에 `AWS_PROFILE` 환경 변수를 확인하여 로그인 공급자에게 전달된 프로필 값을 확인합니다`default`.

SDK for PHP의 **`assumeRole`** 공급자

역할 수임을 통해 `Aws\Credentials\AssumeRoleCredentialProvider`를 사용하여 보안 인증을 생성하는 경우 표시된 것처럼 `'client'` 정보를 `StsClient` 객체와 함께 제공하고 `'assume_role_params'` 세부 정보를 제공해야 합니다.

> **Note**
>
> 모든 API 작업에서 자격 AWS STS 증명을 불필요하게 가져오지 않도록 `memoize` 함수를 사용하여 자격 증명이 만료될 때 자격 증명 자동 새로 고침을 처리할 수 있습니다. 예제는 다음 코드를 참조하세요.

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

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

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

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

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

`'assume_role_params'`에 대한 자세한 내용은 AssumeRole을 참조하세요.

SDK for PHP의 **sso** 공급자

`Aws\Credentials\CredentialProvider::sso`는 Single Sign-On 자격 증명 공급자입니다. 이 공급자를 AWS IAM Identity Center 자격 증명 공급자라고도 합니다.

```
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
]);
```

이름이 지정된 프로필을 사용하는 경우 이전 예제에서 프로필 이름을 `default`로 대체하세요. 이름이 지정된 프로필을 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 공유 *config* 및 *credentials* 파일을 참조하세요. 또는 `AWS_PROFILE` 환경 변수를 사용하여 사용할 프로필 설정을 지정할 수 있습니다.

IAM ID 센터 공급자의 작동 방식을 자세히 알아보려면 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증 이해를 참조하세요.

SDK for PHP의 **`defaultProvider`** 공급자

`Aws\Credentials\CredentialProvider::defaultProvider`는 기본 자격 증명 공급자이며 기본 자격 증명 공급자 체인이라고도 합니다. 이 공급자는 클라이언트를 생성할 때 `credentials` 옵션을 생략한 경우에 사용됩니다. 예를 들어, 다음 코드 스니펫과 같이 S3Client를 생성하면 SDK는 기본 공급자를 사용합니다.

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

체인의 특정 자격 증명 공급자에 파라미터를 제공하려는 경우 코드에서 defaultProvider를 사용할 수도 있습니다. 예를 들어 다음 예제는 `ecsCredentials` 공급자 함수가 사용되는 경우 사용자 지정 연결 제한 시간 및 재시도 설정을 제공합니다.

```
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
]);
```

SDK for PHP의 **ecsCredentials** 공급자

Aws\Credentials\CredentialProvider::ecsCredentials는 GET 요청을 통해 보안 인증을 로드하려고 시도합니다. 이 요청의 URI는 컨테이너에 있는 환경 변수 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI에서 지정됩니다.

```
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
]);
```

SDK for PHP의 **env** 공급자

환경 변수를 사용하여 자격 증명을 포함하면 AWS 실수로 보안 액세스 키를 공유할 수 없습니다. 프로덕션 파일의 클라이언트에 AWS 액세스 키를 직접 추가하지 않는 것이 좋습니다.

SDK는 Amazon Web Services에 인증하기 위해 먼저 환경 변수에서 보안 인증을 확인합니다. SDK는 getenv() 함수를 사용하여 AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY 및 AWS_SESSION_TOKEN 환경 변수를 찾습니다. 이러한 보안 인증을 환경 보안 인증이라고 합니다. 이러한 값을 얻는 방법에 대한 지침은 AWS SDK 및 도구 참조 가이드의 단기 보안 인증을 사용한 인증을 참조하세요.

에서 애플리케이션을 호스팅하는 경우 SDK가 이러한 자격 증명을 자동으로 사용할 AWS Elastic Beanstalk수 있도록 AWS Elastic Beanstalk 콘솔을 통해 AWS_ACCESS_KEY_IDAWS_SECRET_KEY, 및 AWS_SESSION_TOKEN 환경 변수를 설정할 수 있습니다.

환경 변수를 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 환경 변수 지원 참조하세요. 또한 대부분의 AWS SDKs. https://docs.aws.amazon.com/sdkref/latest/guide/settings-reference.html#EVarSettings

다음과 같이 명령줄에서 환경 변수를 설정할 수도 있습니다.

Linux

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

Windows

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

`Aws\Credentials\CredentialProvider::env`는 환경 변수에서 보안 인증을 로드하려고 시도합니다.

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

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

SDK for PHP의 **`assumeRoleWithWebIdentityCredentialProvider`** 공급자

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider`는

역할을 위임하여 보안 인증 로드를 시도합니다. 환경 변수 AWS_ROLE_ARN 및 AWS_WEB_IDENTITY_TOKEN_FILE이 존재할 경우, 공급자는 AWS_WEB_IDENTITY_TOKEN_FILE에 지정된 전체 경로에서 디스크의 토큰을 사용하여 AWS_ROLE_ARN에서 지정된 역할 위임을 시도합니다. 환경 변수를 사용할 경우, 공급자는 AWS_ROLE_SESSION_NAME 환경 변수로부터 세션 설정을 시도합니다.

환경 변수가 설정되어 있지 않다면, 공급자는 기본 프로필 또는 AWS_PROFILE으로 설정된 값을 사용합니다. 공급자는 기본적으로 ~/.aws/credentials 및 ~/.aws/config에서 프로필을 읽으며, filename 컨피그 옵션에서 지정된 프로필을 읽어 들일 수 있습니다. 공급자는 프로필의 role_arn에서 역할을 위임하며, web_identity_token_file에서 설정된 전체 경로로부터 토큰을 읽습니다. role_session_name은 프로필에 설정되어 있는 경우 사용할 수 있습니다.

공급자는 기본 공급망의 한 부분으로서 호출되지만 직접 호출될 수도 있습니다.

```
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
]);
```

기본적으로 이 보안 인증 공급자는 StsClient에서 역할을 위임하기 위해 사용하는 구성된 리전을 상속합니다. 선택적으로 전체 StsClient를 제공할 수 있습니다. 보안 인증은 제공된 모든 StsClient에서 false로 설정해야 합니다.

```
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
]);
```

SDK for PHP의 **ini** 공급자

`Aws\Credentials\CredentialProvider::ini`는 공유 `config` 및 `credentials` 파일에서 자격 증명을 로드하려고 시도합니다. 기본적으로 SDK는에 있는 공유 AWS `credentials` 파일에서 "기본" 프로파일을 로드하려고 시도합니다`~/.aws/credentials`. SDK가 `AWS_SDK_LOAD_NONDEFAULT_CONFIG` 환경 변수를 찾으면에 있는 공유 AWS `config` 파일에서 "기본" 프로필도 확인합니다`~/.aws/config`.

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

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

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

공급자를 생성하는 함수에 인수를 제공하여 사용자 지정 프로파일 또는 .ini 파일 위치를 사용할 수 있습니다.

```
$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
]);
```

SDK for PHP의 **process** 공급자

`Aws\Credentials\CredentialProvider::process`는 공유 AWS 구성 파일의 프로파일에 지정된 `credential_process` 값을 실행하여 자격 증명을 로드하려고 시도합니다.

기본적으로 SDK는 먼저 있는 공유 AWS `credentials` 파일에서 "기본" 프로파일을 로드하려고 시도합니다`~/.aws/credentials`. 공유 `credentials` 파일에서 "기본" 프로필을 찾을 수 없는 경우 공유 `config` 파일에서 기본 프로필을 찾습니다. 다음은 공유 `credentials` 파일의 구성 예시입니다.

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

SDK는 PHP의 `shell_exec` 함수를 사용하여 제공된 그대로 `credential_process` 명령을 호출한 후 stdout에서 JSON 데이터를 읽습니다. `credential_process`는 다음 형식으로 자격 증명을 stdout에 작성해야 합니다.

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

`SessionToken` 및 `Expiration`은 선택 사항 두 선택 사항을 지정한다면 보안 인증이 임시로 처리됩니다.

```
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
]);
```

공급자를 생성하는 함수에 인수를 제공하여 사용자 지정 프로파일 또는 .ini 파일 위치를 사용할 수 있습니다.

```
$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
]);
```

SDK for PHP의 **`instanceProfile`** 공급자

`Aws\Credentials\CredentialProvider::instanceProfile`은 Amazon EC2 인스턴스 프로파일에 지정된 IAM 역할에 대한 자격 증명을 로드하려고 시도합니다.

```
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
]);
```

기본적으로 공급자는 보안 인증을 가져오기 위해 최대 3번까지 재시도할 수 있습니다. 재시도 횟수는 `retries` 옵션으로 설정할 수 있으며, 다음 코드에서와 같이 옵션을 `0`으로 설정하여 완전히 비활성화할 수 있습니다.

```
use Aws\Credentials\CredentialProvider;

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

환경 변수 `AWS_METADATA_SERVICE_NUM_ATTEMPTS`가 설정되어 있는 경우 그 값이 앞서 표시된 'retries' 옵션보다 우선 적용됩니다.

> **Note**
>
> `AWS_EC2_METADATA_DISABLED` 환경 변수를 `true`로 설정하면 Amazon EC2 인스턴스 프로파일에서 로드하려는 이 시도를 비활성화할 수 있습니다.

SDK for PHP에서 자격 증명 공급자 체인화

`Aws\Credentials\CredentialProvider::chain()` 함수를 사용하여 보안 인증 공급자를 연결할 수 있습니다. 이 함수는 variadic 수의 인수를 받으며, 각 인수는 보안 인증 공급자 함수입니다. 따라서 이 함수는 공급자 중 하나가 성공적으로 이행된 promise를 반환할 때까지 함수가 하나씩 차례로 호출되도록 제공된 함수의 합성인 새 함수를 반환합니다.

`defaultProvider`는 이 합성을 사용하여 실패하기 전에 여러 공급자를 확인합니다. `defaultProvider`의 소스는 `chain` 함수의 사용을 보여 줍니다.

```
// This function returns a provider
public static function defaultProvider(array $config = [])
{
    // This function is the provider, which is actually the composition
    // of multiple providers. Notice that we are also memoizing the result by
    // default.
```

```
    return self::memoize(
        self::chain(
            self::env(),
            self::ini(),
            self::instanceProfile($config)
        )
    );
}
```

SDK for PHP와 함께 사용할 사용자 지정 자격 증명 공급자 생성

보안 인증 공급자는 호출될 때 `GuzzleHttp\Promise\PromiseInterface` 객체를 통해 이행되거나 `Aws\Credentials\CredentialsInterface`을 통해 거부되는 promise(`Aws\Exception\CredentialsException`)를 반환하는 함수일 뿐입니다.

공급자를 생성하기 위한 모범 사례는 실제 보안 인증 공급자를 생성하기 위해 호출되는 함수를 생성하는 것입니다. 예를 들어 다음은 env 공급자의 소스입니다(예시용으로 약간 수정됨). 이 소스는 실제 공급자 함수를 반환하는 함수라는 점에 주의하세요. 이 함수를 사용하여 보안 인증 공급자를 쉽게 생성하고 값으로 전달할 수 있습니다.

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

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

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

SDK for PHP에서 자격 증명 메모이제이션

때로는 이전 반환 값을 기억하는 보안 인증 공급자를 생성해야 할 수도 있습니다. 이 공급자는 보안 인증 로딩이 비용이 많이 드는 작업인 경우 또는 `Aws\Sdk` 클래스를 사용하여 여러 클라이언트 간에 보안 인증 공급자를 공유하는 경우 성능에 유용합니다. 보안 인증 공급자 함수를 `memoize` 함수 안에 래핑하여 메모이제이션(memoization)을 보안 인증 공급자에 추가할 수 있습니다.

```
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());
```

메모이제이션(memoization)된 보안 인증이 만료되면 메모이제이션 래퍼가 보안 인증을 새로 고치려고 시도할 때 래핑된 공급자를 호출합니다.

AWS SDK for PHP 버전 3을 사용하여 IAM 역할 수임

Amazon EC2 인스턴스 변수 보안 인증에 IAM 역할 사용

Amazon EC2 인스턴스에서 애플리케이션을 실행 중인 경우 AWS를 호출하기 위해 보안 인증을 제공하는 기본 방법은 IAM 역할을 사용하여 임시 보안 보안 인증을 가져오는 것입니다.

IAM 역할을 사용하는 경우 애플리케이션에서 보안 인증 관리에 대해 걱정할 필요가 없습니다. Amazon EC2 인스턴스의 메타데이터 서버에서 임시 보안 인증을 검색하여 인스턴스가 역할을 “수임”할 수 있습니다.

인스턴스 프로파일 보안 인증이라고도 하는 임시 보안 인증을 사용하면 역할 정책에서 허용하는 작업 및 리소스에 액세스할 수 있습니다. Amazon EC2는 IAM 서비스에 인스턴스를 안전하게 인증하여 역할을 수임하고, 검색된 역할 보안 인증을 정기적으로 업데이트하는 모든 작업을 처리합니다. 이러한 방

식으로 사용자의 별도 작업 없이 애플리케이션을 안전하게 보호할 수 있습니다. 임시 보안 인증 정보를 지원하는 서비스 목록은 IAM 사용 설명서의 IAM으로 작업하는 AWS를 참조하세요.

> **Note**
>
> 매번 메타데이터 서비스를 실행하지 않으려면 `Aws\CacheInterface`의 인스턴스를 `'credentials'` 옵션으로 클라이언트 생성자에 전달할 수 있습니다. 이렇게 하면 SDK가 캐시된 인스턴스 프로파일 보안 인증을 대신 사용할 수 있습니다. 자세한 내용은 AWS SDK for PHP 버전 3 구성을 참조하세요.

SDK를 사용하여 Amazon EC2 애플리케이션을 개발하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 Amazon EC2 인스턴스용 IAM 역할 사용을 참조하세요.

IAM 역할을 생성 및 Amazon EC2 인스턴스에 할당

1. IAM 클라이언트 생성

   가져옵니다.

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

   use Aws\Iam\IamClient;
   ```

   샘플 코드

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

2. 사용할 작업과 리소스에 필요한 권한으로 IAM 역할을 생성합니다.

   샘플 코드

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

3. IAM 인스턴스 프로파일을 생성하고 결과에서 Amazon 리소스 이름(ARN)을 생성합니다.

> **Note**
>
> AWS SDK for PHP 대신에 IAM 콘솔을 사용하는 경우 이 콘솔은 인스턴스 프로파일을 자동으로 생성한 후 해당하는 역할과 동일한 이름을 이 프로파일에 지정합니다.
> 샘플 코드
>
> ```
> $IPN = 'InstanceProfileName';
>
> $result = $client->createInstanceProfile([
>     'InstanceProfileName' => $IPN ,
> ]);
>
> $ARN = $result['Arn'];
> $InstanceID =  $result['InstanceProfileId'];
> ```

4. Amazon EC2 클라이언트 생성

   가져옵니다.

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

   use Aws\Ec2\Ec2Client;
   ```

   샘플 코드

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

5. Amazon EC2 인스턴스 프로파일을 실행 중이거나 중지된 Amazon EC2 인스턴에 추가합니다. IAM 역할의 인스턴스 프로파일 이름을 사용합니다.

   샘플 코드

   ```
   $result = $ec2Client->associateIamInstanceProfile([
      'IamInstanceProfile' => [
          'Arn' => $ARN,
   ```

```
            'Name' => $IPN,
        ],
        'InstanceId' => $InstanceID
]);
```

자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2의 IAM 역할을 참조하세요.

Amazon ECS 태스크에 대한 IAM 역할 사용하기

Amazon Elastic Container Service(Amazon ECS) 작업의 경우 IAM 역할을 맡아 AWS API 호출을 수행할 수 있습니다. 이 기능은 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 보안 인증을 제공하는 것과 비슷한 방식으로 애플리케이션에서 보안 인증을 관리할 수 있는 전략을 제공합니다.

AWS 보안 인증을 생성하여 컨테이너에 배포하거나 Amazon EC2 인스턴스의 역할을 사용하는 대신, IAM 역할을 ECS 작업 정의 또는 `RunTask` API 작업과 연결할 수 있습니다.

컨테이너 작업이 맡을 수 있는 IAM 역할을 사용하는 방법에 대한 자세한 내용은 Amazon ECS 개발자 안내서의 작업 IAM 역할 주제를 참조하세요. 작업 정의에서 작업 IAM 역할을 a `taskRoleArn` 형식으로 사용하는 예제는 Amazon ECS 개발자 안내서에 있는 예제 작업 정의를 참조하세요.

다른 AWS 계정에서 IAM 역할을 맡는 경우

AWS 계정 계정(계정 A)에서 작업을 수행하며 다른 계정(계정 B)으로 역할을 수임하려는 경우 먼저 계정 B로 IAM 역할을 생성해야 합니다. 이 역할이 있으면 계정(계정 A)의 엔티티가 계정 B로 특정 작업을 수행할 수 있습니다. 크로스 계정 액세스에 대한 자세한 내용은 자습서: IAM 역할을 사용한 AWS 크로스 계정 액세스 권한 위임 단원AWS을 참조하세요.

계정 B로 역할을 생성한 후 역할 ARN을 기록합니다. 이 ARN은 계정 A의 역할을 수임할 때 사용합니다. 계정 A의 엔티티와 연결된 AWS 보안 인증을 사용하여 역할을 수임합니다.

AWS 계정 계정에 대한 보안 인증을 사용하여 AWS STS 클라이언트를 생성합니다. 다음에서는 보안 인증 프로파일을 사용했지만 어떤 방법이든 사용할 수 있습니다. 새로 생성된 AWS STS 클라이언트를 사용하여 assume-role을 호출하고 사용자 지정 sessionName을 제공합니다. 결과에서 새 임시 보안 인증을 검색합니다. 기본적으로 보안 인증은 1시간 지속됩니다.

샘플 코드

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

자세한 내용은 AWS SDK for PHP API 참조의 IAM 역할 사용하기 또는 AssumeRole을 참조하세요.

웹 보안 인증으로 IAM 역할 사용

웹 ID 페더레이션을 통해 고객이 AWS 리소스에 액세스할 때 인증하는 데 타사 보안 인증 공급자를 사용할 수 있습니다. 웹 보안 인증을 사용하여 역할을 수임하려면 먼저 IAM 역할을 생성하고 웹 ID 제공업체(idP)를 구성해야 합니다. 자세한 내용은 웹 ID 또는 OpenID Connect 페더레이션을 위한 역할 생성(콘솔)을 참조하세요.

ID 제공업체를 생성하고 웹 ID를 위한 역할을 생성한 후 AWS STS 클라이언트를 사용하여 사용자를 인증합니다. 해당 사용자에 대한 권한으로 보안 인증에 대한 webIdentityToken 및 ProviderId를 제공하고 IAM 역할에 대한 역할 ARN을 제공합니다.

샘플 코드

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

자세한 내용은 AWS SDK for PHP API 참조의 AssumeRoleWithWebIdentity — 웹 기반 ID 공급자를 통한 페더레이션 또는 AssumeRoleWithWebIdentity를 참조하세요.

프로필을 사용하여 역할 수임

~/.aws/credentials에서 프로필을 정의하세요

~/.aws/credentials 파일에서 역할에 대한 프로파일을 정의하여 IAM 역할을 사용하도록 AWS SDK for PHP을 구성할 수 있습니다.

수임할 역할에 대해 role_arn 설정을 사용하여 새 프로필을 생성합니다. 또한 IAM 역할을 수임할 수 있는 권한을 가진 보안 인증을 사용하여 다른 프로필의 source_profile 설정을 포함합니다. 이러한 구성 설정에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 역할 보안 인증 수임을 참조하세요.

예를 들어, 다음 ~/.aws/credentials에서 project1 프로필은 role_arn을 설정하고 default 프로필을 보안 인증의 원본으로 지정하여 연결된 엔티티가 역할을 수임할 수 있는지 확인합니다.

```
[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
```

클라이언트를 시작할 때 AWS_PROFILE 환경 변수 또는 profile 파라미터를 설정하여 default 프로필을 소스 보안 인증으로 사용하면 project1에 지정된 역할이 수임됩니다.

다음 S3Client 스니펫은 생성자에서 profile 파라미터를 사용하는 방법을 보여줍니다. S3Client에는 project1 프로필과 관련된 역할과 관련된 권한이 있습니다.

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

~/.aws/config에서 프로필을 정의하세요

이 ~/.aws/config 파일에는 가정하려는 프로필도 포함될 수 있습니다. 환경 변수를 AWS_SDK_LOAD_NONDEFAULT_CONFIG로 설정하면 config 파일에서 프로파일을 로드합니다. AWS_SDK_LOAD_NONDEFAULT_CONFIG를 설정하면 SDK는 ~/.aws/config 및 ~/.aws/credentials 모두에서 프로필을 로드합니다. ~/.aws/credentials의 프로필이 마지막으로 로드되고 ~/.aws/config에 있는 동일한 이름의 프로필보다 우선 적용됩니다. 어느 위치의 프로필이든 source_profile 또는 수임할 프로필로 사용할 수 있습니다.

다음 예제에서는 파일에 정의된 config 파일의 project1 프로필과 credentials파일의 default 프로필을 사용합니다. AWS_SDK_LOAD_NONDEFAULT_CONFIG도 설정됩니다.

```
# 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
```

다음 스니펫과 같이 `S3Client` 생성자가 실행되면 프로필에 정의된 역할은 `project1` 프로필과 `default` 관련된 보안 인증을 사용하여 가정됩니다.

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

SDK for PHP에서 AWS STS의 임시 자격 증명 사용

AWS Security Token Service(AWS STS)를 사용하면 ID 페더레이션을 통해 IAM 사용자 또는 인증한 사용자에 대해 제한된 권한, 임시 보안 인증을 요청할 수 있습니다. 자세한 내용은 IAM 사용자 설명서의 임시 보안 보안 인증을 참조하세요. 임시 보안 보안 인증을 사용해 대부분의 AWS 서비스에 액세스할 수 있습니다. 임시 보안 인증 정보를 지원하는 서비스 목록은 IAM 사용자 설명서의 IAM으로 작업하는 AWS를 참조하세요.

임시 보안 인증에 대한 일반 사용 사례 중 하나는 타사 보안 인증 공급자를 통해 사용자를 인증하여 모바일 또는 클라이언트 측 애플리케이션 액세스 권한을 AWS 리소스에 부여하는 것입니다(웹 ID 페더레이션 참조).

임시 보안 인증 얻기

AWS STS에는 임시 보안 인증을 반환하는 여러 작업이 있지만, `GetSessionToken` 작업은 데모용으로 가장 간단한 작업입니다. 다음 스니펫은 PHP SDK STS 클라이언트의 `getSessionToken` 메서드를 호출하여 임시 자격 증명을 검색합니다.

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

$stsClient = $sdk->createSts();

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

`GetSessionToken` 및 기타 AWS STS 작업의 결과에는 항상 `'Credentials'` 값이 포함됩니다. `$result`를 인쇄하면(예: `print_r($result)` 사용) 다음과 같습니다.

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

AWS SDK for PHP에 임시 보안 인증 제공

클라이언트를 인스턴스화하고 AWS STS에서 직접 수신된 값을 전달하여 임시 보안 인증을 다른 AWS 클라이언트와 함께 사용할 수 있습니다.

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

또한 `Aws\Credentials\Credentials` 객체를 생성하고 클라이언트를 인스턴스화할 때 해당 객체를 사용할 수도 있습니다.

```
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
]);
```

하지만 임시 보안 인증을 제공하는 가장 좋은 방법은 `StsClient`와 함께 포함된 `createCredentials()` 헬퍼 메서드를 사용하는 것입니다. 이 메서드는 AWS STS 결과에서 데이터를 추출하고 `Credentials` 객체를 생성합니다.

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

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

애플리케이션 또는 프로젝트에서 임시 보안 인증을 사용해야 하는 이유에 대한 자세한 내용은 AWS STS 설명서의 임시 액세스 허용 시나리오를 참조하세요.

SDK for PHP에서 익명 클라이언트 생성

보안 인증과 연결되지 않은 클라이언트를 만들어야 할 경우가 있습니다. 이렇게 하면 서비스에 익명 요청을 수행할 수 있습니다.

예를 들어 익명 액세스를 허용하도록 Amazon S3 객체와 Amazon CloudSearch 도메인을 둘 다 구성할 수 있습니다.

익명 클라이언트를 생성하려면 `'credentials'` 옵션을 `false`로 설정할 수 있습니다.

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

```
// Makes an anonymous request. The object would need to be publicly
// readable for this to succeed.
$result = $s3Client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
]);
```

SDK for PHP에서 AWS 공통 런타임(AWS CRT) 확장 사용

AWS CRT 라이브러리는 여러 AWS SDKs에 대해 우수한 성능과 최소한의 공간으로 기본 기능을 제공합니다. 이 주제에서는 SDK for PHP에서 AWS CRT를 사용하는 시기와 AWS CRT 확장을 설치하는 방법을 설명합니다.

AWS CRT 확장을 설치해야 하는 경우

PHP용 SDK는 AWS CRT 라이브러리의 권한 부여 및 체크섬 기능을 사용합니다. AWS CRT 확장은 다음 작업을 수행할 때 필요합니다.

- Amazon S3 다중 리전 액세스 포인트
- Amazon EventBridge 글로벌 엔드포인트
- Amazon Simple Storage Service(Amazon S3)의 CRC-32C 체크섬 알고리즘

위에 나열된 기능을 사용하고 PHP 환경에 AWS CRT 확장이 설치되지 않은 경우 PHP용 SDK는 오류 메시지를 보고하고 확장을 설치하라는 메시지를 표시합니다.

AWS 공통 런타임(AWS CRT) 확장 설치

AWS CRT 확장을 설치하는 방법에 대한 지침은 aws-crt-php용 GitHub 리포지토리의 기본 페이지에서 확인할 수 있습니다.

AWS SDK for PHP 버전 3 사용

이 장에서는 AWS SDK for PHP 버전 3을 사용하여 AWS 서비스와 효과적으로 상호 작용하는 방법에 대한 포괄적인 지침을 제공합니다. 서비스 요청 수행, 오류 처리, 고급 SDK 기능 활용을 위한 핵심 기술을 학습하여 AWS에서 견고한 PHP 애플리케이션을 구축하는 방법을 알아보게 됩니다.

간단한 애플리케이션이든 복잡한 시스템이든, 이 장에서 설명하는 기술들은 PHP SDK를 통해 AWS 서비스와 작업할 때 코드를 최적화하고, 오류 처리를 개선하며, 효율적인 패턴을 구현하는 데 도움이 될 것입니다.

주제

AWS SDK for PHP 버전 3을 사용하여 AWS 서비스 요청하기

SDK 요청 워크플로 개요

AWS SDK for PHP 버전 3 작업은 모든 AWS 서비스에서 일관된 패턴을 따릅니다. 기본 워크플로는 세 가지 주요 단계로 구성됩니다.

1. 서비스 클라이언트 생성 - 사용하려는 AWS 서비스에 대한 클라이언트 객체를 인스턴스화합니다.
2. 작업 실행 - 서비스 API의 작업에 해당하는 클라이언트 메서드를 호출합니다.
3. 결과 처리 - 성공 시 반환된 배열형 결과 객체를 처리하거나, 실패 시 발생하는 예외를 처리합니다.

다음 섹션에서는 서비스 클라이언트를 생성하고 구성하는 방법부터 시작하여 이러한 각 단계를 자세히 설명합니다.

기본 서비스 클라이언트 생성

결합형 배열의 옵션을 클라이언트의 생성자에 전달하여 클라이언트를 만들 수 있습니다.

가져옵니다.

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

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

샘플 코드

```
//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'.
```

선택적 'version' 파라미터에 대한 정보는 구성 옵션 항목에서 확인할 수 있습니다.

보안 인증을 클라이언트에 명시적으로 제공하지 않았다는 점에 주의하세요. 이는 SDK가 기본 자격 증명 공급자 체인을 사용하여 자격 증명 정보를 찾기 때문입니다.

모든 일반 클라이언트 구성 옵션에 대해서는 클라이언트 생성자 옵션(AWS SDK for PHP 버전 3)에서 자세히 설명합니다. 클라이언트에 제공되는 옵션의 배열은 어떤 클라이언트를 생성하고 있는지에 따라 다를 수 있습니다. 이러한 사용자 지정 클라이언트 구성 옵션에 대해서는 각 클라이언트의 API 설명서에서 설명합니다.

위의 예시는 기본적인 클라이언트 생성을 보여줍니다. 그러나 특정 요구 사항을 충족시키기 위해 서비스 클라이언트를 사용자 지정할 수 있습니다. 코드를 통한 서비스 클라이언트 구성에 대한 자세한 내용은 코드 내에서 AWS SDK for PHP 버전 3용 서비스 클라이언트 구성 섹션을 참조하세요. 외부 구성 파일 또는 환경 변수를 사용하여 서비스 클라이언트를 구성해야 하는 경우 외부에서 AWS SDK for PHP 버전 3용 서비스 클라이언트 구성 섹션을 참조하세요.

요청하기

클라이언트 객체에서 동일한 이름의 메서드를 호출하여 서비스 요청을 수행할 수 있습니다. 예를 들어, Amazon S3 PutObject 작업을 수행하려면 `Aws\S3\S3Client::putObject()` 메서드를 호출합니다.

가져옵니다.

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

use Aws\S3\S3Client;
```

샘플 코드

```
// 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'];
```

클라이언트에 사용할 수 있는 작업과 입력 및 출력의 구조는 서비스 설명 파일을 기반으로 실행 시간에 정의됩니다. 클라이언트를 생성할 때 서비스 모델의 `version` 파라미터(예: “2006-03-01” 또는 “latest”)를 제공하지 않으면 클라이언트는 기본적으로 최신 버전으로 설정됩니다. SDK는 제공된 버전을 기반으로 해당 구성 파일을 찾습니다.

putObject()와 같은 작업 메서드는 모두 작업의 파라미터를 나타내는 결합형 배열인 단일 인수를 받습니다. 이 배열의 구조(및 결과 객체의 구조)는 SDK의 API 설명서에서 각 작업에 대해 정의됩니다(예: putObject 작업에 대한 API 설명서 참조).

HTTP 핸들러 옵션

특정 @http 파라미터를 사용하여 기본 HTTP 핸들러가 요청을 실행하는 방법도 미세 조정할 수 있습니다. @http 파라미터에 포함시킬 수 있는 옵션은 "http" 클라이언트 옵션을 사용하여 클라이언트를 인스턴스화할 때 설정할 수 있는 옵션과 동일합니다.

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

결과 객체 작업

성공적인 작업을 실행하면 Aws\Result 객체가 반환됩니다. 서비스의 원시 XML 또는 JSON 데이터를 반환하는 대신, SDK는 응답 데이터를 결합형 배열 구조로 강제 변환합니다. SDK는 특정 서비스 및 기본 응답 구조에 대한 지식을 기반으로 데이터의 일부 측면을 정규화합니다.

AWS\Result 객체에서 결합형 PHP 배열처럼 데이터에 액세스할 수 있습니다.

가져옵니다.

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

샘플 코드

```
// 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();
```

결과 객체의 콘텐츠는 실행되는 작업과 서비스의 버전에 따라 다릅니다. 각 API의 결과 구조는 각 작업에 대한 API 설명서에서 문서화됩니다.

SDK는 JSON 데이터 또는 우리의 경우 PHP 배열을 검색하고 조작하는 데 사용되는 DSL인 JMesPath와 통합됩니다. 결과 객체에는 결과에서 데이터를 더 선언적으로 추출하기 위해 사용할 수 있는 `search()` 메서드가 포함됩니다.

샘플 코드

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

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

AWS SDK for PHP 버전 3의 커맨드 오브젝트

AWS SDK for PHP는 명령 패턴을 사용하여 나중에 HTTP 요청을 전송하는 데 사용될 파라미터와 핸들러를 캡슐화합니다.

명령의 암시적 사용

클라이언트 클래스를 검사하면 API 작업에 해당하는 메서드가 실제로 존재하지 않는 것을 확인할 수 있습니다. 이러한 메서드는 `__call()` magic 메서드를 사용하여 구현됩니다. 이러한 의사 메서드는 실제로 SDK의 명령 객체 사용을 캡슐화하는 바로 가기입니다.

일반적으로 명령 객체와 직접 상호 작용할 필요는 없습니다. `Aws\S3\S3Client::putObject()`와 같은 메서드를 호출하면 SDK는 제공된 파라미터를 기반으로 `Aws\CommandInterface` 객체를 실제

로 생성하고, 명령을 실행한 다음, 채워진 `Aws\ResultInterface` 객체를 반환합니다(또는 예외나 오류 발생). 클라이언트의 `Async` 메서드 중 하나(예: `Aws\S3\S3Client::putObjectAsync()`)를 호출하면 비슷한 워크플로가 수행됩니다. 클라이언트는 제공된 파라미터를 기반으로 명령을 생성하고, HTTP 요청을 직렬화한 다음, 요청을 시작하고, promise를 반환합니다.

다음은 기능적으로 동등한 예제입니다.

```
$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);
```

명령 파라미터

모든 명령은 서비스의 API에 속하지 않지만 그 대신 SDK의 동작을 제어하는 몇 가지 특수 파라미터를 지원합니다.

`@http`

이 파라미터를 사용하여 기본 HTTP 핸들러가 요청을 실행하는 방식을 미세 조정할 수 있습니다. `@http` 파라미터에 포함시킬 수 있는 옵션은 "http" 클라이언트 옵션을 사용하여 클라이언트를 인스턴스화할 때 설정할 수 있는 옵션과 동일합니다.

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

`@retries`

"재시도" 클라이언트 옵션과 마찬가지로, `@retries`는 실패한 것으로 간주되기 전에 명령을 재시도할 수 있는 횟수를 제어합니다. 재시도를 비활성화하려면 이 옵션을 `0`으로 설정합니다.

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

> **Note**
>
> 클라이언트에서 재시도를 비활성화한 경우 해당 클라이언트에 전달된 개별 명령에서 재시도를 선택적으로 활성화할 수 없습니다.

명령 객체 생성

클라이언트의 `getCommand()` 메서드를 사용하여 명령을 생성할 수 있습니다. 이 명령은 HTTP 요청을 즉시 실행하거나 전송하지 않으며, 클라이언트의 `execute()` 메서드에 전달될 때만 실행됩니다. 따라서 명령을 실행하기 전에 명령 객체를 수정할 기회가 있습니다.

```
$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);
```

명령 `HandlerList`

클라이언트에서 명령을 생성한 경우 클라이언트 `Aws\HandlerList` 객체의 복제가 명령에 제공됩니다. 명령이 클라이언트에서 실행하는 다른 명령에 영향을 미치지 않는 사용자 지정 미들웨어와 핸들러를 사용할 수 있도록 클라이언트 핸들러 목록의 복제가 명령에 제공됩니다.

따라서 명령별로 다른 HTTP 클라이언트를 사용할 수 있으며(예: Aws\MockHandler) 미들웨어를 통해 명령별로 사용자 지정 동작을 추가할 수 있습니다. 다음 예제에서는 실제 HTTP 요청을 전송하는 대신 MockHandler를 사용하여 의사 결과를 생성합니다.

```
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'
```

명령에 사용되는 핸들러를 변경할 수 있을 뿐 아니라, 사용자 지정 미들웨어를 명령에 주입할 수도 있습니다. 다음 예제에서는 핸들러 목록에서 관찰자로 작동하는 tap 미들웨어를 사용합니다.

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

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

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

// Append the middleware to the "sign" step of the handler list. The sign
// step is the last step before transferring an HTTP request.
```

```
$list->append('sign', $middleware);

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

CommandPool

Aws\CommandPool을 사용하면 Aws\CommandInterface 객체를 산출하는 반복자를 사용하여 명령을 동시에 실행할 수 있습니다. CommandPool은 풀의 명령을 반복하는 동안 일정한 수의 명령이 동시에 실행되도록 보장합니다(명령이 완료되면 일정한 풀 크기를 유지하기 위해 추가 명령이 실행됨).

다음은 CommandPool을 사용하여 몇 개의 명령만 전송하는 매우 간단한 예제입니다.

```
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();
```

이 예제는 CommandPool의 성능을 상당히 낮춘 것입니다. 더 복잡한 예제를 살펴보겠습니다. 디스크의 파일을 Amazon S3 버킷에 업로드한다고 가정합니다. 디스크에서 파일 목록을 가져오려면 PHP의 DirectoryIterator를 사용할 수 있습니다. 이 반복자는 SplFileInfo 객체를 생성합니다. CommandPool은 Aws\CommandInterface 객체를 산출하는 반복자를 받으므로, SplFileInfo 객체를 반환하도록 Aws\CommandInterface 객체를 매핑합니다.

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

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

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

$fromDir = '/path/to/dir';
$toBucket = 'amzn-s3-demo-bucket';

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

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

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

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

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

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

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

CommandPool 구성

Aws\CommandPool 생성자는 다양한 구성 옵션을 받습니다.

concurrency (callable|int)

동시에 실행할 최대 명령 수입니다. 동적으로 풀 크기를 조정하려면 함수를 제공합니다. 함수에는 현재 보류 중인 요청 수가 제공되며 이 함수는 새 풀 크기 한도를 나타내는 정수를 반환할 것으로 예상됩니다.

before (callable)

각 명령을 전송하기 전에 호출할 함수입니다. `before` 함수는 명령과 명령의 반복자 키를 받습니다. 명령을 전송하기 전에 `before` 함수에서 필요에 따라 명령을 변형할 수 있습니다.

fulfilled (callable)

promise가 이행될 때 호출할 함수입니다. 결과 객체, 결과가 나온 반복자의 ID, 풀을 단락시켜야 하는 경우 해결하거나 거부할 수 있는 집계 promise가 함수에 제공됩니다.

rejected (callable)

promise가 거부될 때 호출할 함수입니다. `Aws\Exception` 객체, 예외가 나온 반복자의 ID, 풀을 단락시켜야 하는 경우 해결하거나 거부할 수 있는 집계 promise가 함수에 제공됩니다.

명령 간 수동 가비지 수집

대용량 명령 풀로 인해 메모리 제한에 도달한 경우에는 메모리 제한에 도달했을 때 PHP 가비지 수집기에서 수집된 순환 참조가 아닌, SDK에서 생성된 순환 참조가 원인일 수 있습니다. 이때는 명령 사이에 수집 알고리즘을 직접 호출하면 제한에 도달하기 전에 순환 참조를 수집할 수 있습니다. 다음 예제는 각 명령을 전송하기 전에 콜백을 사용해 수집 알고리즘을 호출하는 `CommandPool`을 생성하는 것입니다. 단, 가비지 수집기를 호출할 경우 성능 비용이 발생하므로 사용 사례와 환경에 따라 사용하는 것이 좋습니다.

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

AWS SDK for PHP 버전 3을 사용한 비동기 프로그래밍

SDK의 비동기 기능을 사용하여 명령을 동시에 전송할 수 있습니다. 작업 이름에 `Async`라는 접미사를 붙여 요청을 비동기적으로 전송할 수 있습니다. 그러면 요청이 시작되고 promise가 반환됩니다.

promise는 성공 시 결과 객체를 통해 이행되거나 실패 시 예외를 통해 거부됩니다. 이러한 방식으로 여러 promise를 생성하여 기본 HTTP 핸들러가 요청을 전송할 때 promise가 HTTP 요청을 동시에 전송하도록 할 수 있습니다.

가져옵니다.

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

샘플 코드

```
// 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();
```

promise의 `wait` 메서드를 사용하여 promise를 동기적으로 강제 완료할 수 있습니다. promise를 강제 완료하면 기본적으로 promise의 상태도 “언래핑” 되므로, promise의 결과를 반환하거나 발생한 예외를 발생시킵니다. promise에서 `wait()`를 호출하면 HTTP 요청이 완료되고 결과가 채워지거나 예외가 발생할 때까지 프로세스가 차단됩니다.

이벤트 루프 라이브러리와 함께 SDK를 사용할 때는 결과에 대해 차단하지 마세요. 그 대신, 결과의 `then()` 메서드를 사용하여 작업이 완료될 때 해결되거나 거부되는 promise에 액세스합니다.

가져오기

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

샘플 코드

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

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

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

AWS SDK for PHP 버전 3에서의 Promise

AWS SDK for PHP는 promise를 사용하여 비동기 워크플로를 허용하며, 이 비동기성 덕분에 HTTP 요청을 동시에 전송할 수 있습니다. SDK에서 사용하는 promise 사양은 Promises/A+입니다.

Promise란 무엇입니까?

promise는 비동기 작업의 최종 결과를 나타냅니다. promise와 상호 작용하는 기본 방법은 `then` 메서드를 통하는 것입니다. 이 메서드는 promise의 최종 값 또는 promise를 이행할 수 없는 이유를 수신할 콜백을 등록합니다.

AWS SDK for PHP는 promise 구현을 위해 guzzlehttp/promises Composer 패키지를 이용합니다. Guzzle promise는 차단 및 비차단을 지원하며 비차단 이벤트 루프와 함께 사용할 수 있습니다.

> **Note**
>
> HTTP 요청은 단일 스레드를 사용하여 AWS SDK for PHP에서 동시에 전송됩니다. 이 경우 상태 변경(예: promise 이행 또는 거부)에 응답하면서 하나 이상의 HTTP 요청을 전송하기 위해 비차단 호출이 사용됩니다.

SDK의 Promise

Promise는 SDK 전체에서 사용됩니다. 예를 들어 페이지네이터, waiter, 명령 풀, 멀티파트 업로드, S3 디렉터리/버킷 전송 등 SDK에서 제공하는 대부분의 상위 수준 추상화에서 프로미스가 사용됩니다.

Async 접미사가 있는 메서드를 호출하면 SDK가 제공하는 모든 클라이언트는 promise를 반환합니다. 예를 들어, 다음 코드는 Amazon DynamoDBDescribeTable 작업의 결과를 가져오기 위해 promise를 생성하는 방법을 보여 줍니다.

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

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

describeTable 또는 describeTableAsync를 호출할 수 있습니다. 이러한 메서드는 클라이언트와 연결된 API 모델 및 __call 번호로 구동되는 클라이언트의 magic version 메서드입니다. describeTable 접미사 없이 Async과 같은 메서드를 호출하면 클라이언트는 HTTP 요청을 전송하는 동안 차단하며 Aws\ResultInterface 객체를 반환하거나 Aws\Exception\AwsException을 발생시킵니다. 작업 이름에 Async(예: describeTableAsync) 접미사를 붙이면 클라이언트는 결국 Aws\ResultInterface 객체를 통해 이행되거나 Aws\Exception\AwsException을 통해 거부되는 promise를 생성합니다.

> **Important**
>
> promise가 반환될 때 결과가 이미 도착했거나(예: 모의(mock) 핸들러 사용 시) HTTP 요청이 시작되지 않았을 수 있습니다.

then 메서드를 사용하여 콜백을 promise에 등록할 수 있습니다. 이 메서드는 모두 선택적인 $onFulfilled 및 $onRejected라는 두 개의 콜백을 받습니다. $onFulfilled 콜백은 promise가 이행되는 경우에 호출되고 $onRejected 콜백은 promise가 거부되는 경우(즉 실패한 경우)에 호출됩니다.

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

동시에 명령 실행

여러 개의 promise를 동시에 실행하도록 함께 작성할 수 있습니다. SDK를 비차단 이벤트 루프와 통합하거나 여러 promise를 빌드하고 이러한 promise가 동시에 완료될 때까지 대기하여 이렇게 할 수 있습니다.

```
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**
>
> CommandPool은 여러 API 작업을 동시에 실행하기 위한 더욱 강력한 메커니즘을 제공합니다.

Promise 연결

promise의 가장 좋은 측면 중 하나는 작성 가능하기 때문에 변환 파이프라인을 생성할 수 있다는 것입니다. Promise는 `then` 콜백을 후속 `then` 콜백과 연결하여 작성됩니다. `then` 메서드의 반환 값은 제공된 콜백의 결과를 기반으로 이행되거나 거부되는 promise입니다.

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

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

> **Note**
>
> promise 콜백의 반환 값은 다운스트림 promise에 제공되는 `$value` 인수입니다. 다운스트림 promise 체인에 값을 제공하려는 경우 콜백 함수에서 값을 반환해야 합니다.

거부 전송

promise가 거부될 때 호출할 콜백을 등록합니다. 콜백에서 예외가 발생하면 promise가 예외를 통해 거부되고 체인의 다음 promise가 예외를 통해 거부됩니다. `$onRejected` 콜백에서 값을 성공적으로 반환하면 `$onRejected` 콜백의 반환 값을 사용하여 promise 체인의 다음 promise가 이행됩니다.

Promise 대기

promise의 `wait` 메서드를 사용하여 promise를 동기적으로 강제 완료할 수 있습니다.

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

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

promise의 `wait` 함수를 호출하는 동안 예외가 발생하면 promise가 예외를 통해 거부되고 예외가 발생합니다.

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

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

이행되지 않은 promise에 대해 `wait`를 호출하면 wait 함수가 트리거되지 않습니다. 이전에 전달한 값만 반환됩니다.

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

거부된 promise에 대해 `wait`를 호출하면 예외가 발생합니다. 거부 이유가 `\Exception`의 인스턴스인 경우 이유가 발생합니다. 그렇지 않으면 `GuzzleHttp\Promise\RejectionException`이 발생하고 예외의 `getReason` 메서드를 호출하여 이유를 얻을 수 있습니다.

> **Note**
>
> AWS SDK for PHP의 API 작업 호출은 `Aws\Exception\AwsException` 클래스의 하위 클래스를 통해 거부됩니다. 하지만 거부 이유를 변경하는 사용자 지정 미들웨어 때문에 `then` 메서드에 전달된 이유가 다를 수 있습니다.

Promise 취소

promise의 `cancel()` 메서드를 사용하여 promise를 취소할 수 있습니다. promise가 이미 해결된 경우 `cancel()`을 호출해도 효과가 없습니다. 한 promise를 취소하면 해당 promise와 해당 promise에서 전달을 대기하고 있는 모든 promise가 취소됩니다. 취소된 promise는 `GuzzleHttp\Promise\RejectionException`을 통해 거부됩니다.

Promise 결합

promise를 집계 promise로 결합하여 더욱 정교한 워크플로를 빌드할 수 있습니다. `guzzlehttp/promise` 패키지에는 promise를 결합하는 데 사용할 수 있는 다양한 함수가 포함되어 있습니다.

namespace-GuzzleHttp.Promise에서 모든 promise 컬렉션 함수에 대한 API 설명서를 참조할 수 있습니다.

each 및 each_limit

고정된 풀 크기로 동시에 수행할 `Aws\CommandInterface` 명령이 포함된 작업 대기열이 있는 경우 CommandPool을 사용하세요(명령은 메모리에 있거나 지연 반복기에 의해 생성될 수 있음). `CommandPool`을 사용하면 제공된 반복자가 소진될 때까지 고정된 수의 명령이 동시에 전송됩니다.

`CommandPool`은 동일한 클라이언트가 실행하는 명령에 대해서만 작동합니다. `GuzzleHttp\Promise\each_limit` 함수를 사용하여 다른 클라이언트의 전송 명령을 고정된 풀 크기로 동시에 수행할 수 있습니다.

```
use GuzzleHttp\Promise;

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

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

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

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

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

Promise 코루틴

Guzzle promise 라이브러리의 더욱 강력한 기능 중 하나는 비동기 워크플로 쓰기를 기존 방식의 동기 워크플로 쓰기와 비슷하게 만드는 promise 코루틴을 사용할 수 있다는 것입니다. 실제로 AWS SDK for PHP는 대부분의 상위 수준 추상화에서 코루틴 promise를 사용합니다.

여러 개의 버킷을 생성하고 버킷이 사용 가능하게 되면 버킷에 파일을 업로드하려고 하며, 이 모든 작업이 최대한 빠르게 수행되도록 모두 동시에 수행하려는 경우를 가정해 봅니다. `all()` promise 함수를 사용하여 여러 개의 코루틴 promise를 함께 결합하면 이 작업을 쉽게 수행할 수 있습니다.

```
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();
```

AWS SDK for PHP 버전 3의 오류 처리

동기 오류 처리

작업을 수행하는 동안 오류가 발생할 경우 예외가 발생합니다. 이러한 이유로 코드에서 오류를 처리해야 하는 경우 작업 둘레에 try/catch 블록을 사용해야 합니다. 오류가 발생하면 SDK는 서비스별 예외를 발생시킵니다.

다음 예제에서는 Aws\S3\S3Client를 사용합니다. 오류가 있는 경우 발생한 예외는 Aws\S3\Exception\S3Exception 유형입니다. SDK가 발생시키는 모든 서비스별 예외는 Aws\Exception\AwsException 클래스에서 확장됩니다. 이 클래스에는 request-id, 오류 코드 및 오류 유형을 포함하여 실패에 대한 유용한 정보가 포함되어 있습니다. 이를 지원하는 몇 가지 서비스를 참고하고, 응답 데이터는 결합형 배열 구조(Aws\Result 객체와 유사함)로 강제 변환되며, 이는 일반 PHP 결합형 배열처럼 액세스할 수 있습니다. toArray() 메서드는 존재할 경우 어떤 데이터이든 반환합니다.

가져오기

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

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

샘플 코드

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

비동기 오류 처리

비동기 요청을 전송할 때는 예외가 발생하지 않습니다. 그 대신, 반환된 promise의 `then()` 또는 `otherwise()` 메서드를 사용하여 결과 또는 오류를 수신해야 합니다.

가져옵니다.

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

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

샘플 코드

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

그 대신 promise를 "언래핑"하고 예외를 발생시킬 수 있습니다.

가져옵니다.

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

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

샘플 코드

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

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

AWS SDK for PHP 버전 3의 핸들러 및 미들웨어

AWS SDK for PHP를 확장하는 기본 메커니즘은 핸들러 및 미들웨어를 사용하는 것입니다. 각 SDK 클라이언트 클래스에는 클라이언트의 `Aws\HandlerList` 메서드를 통해 액세스할 수 있는 `getHandlerList()` 인스턴스가 있습니다. 클라이언트의 `HandlerList`를 검색한 후 클라이언트 동작을 추가하거나 제거하도록 수정할 수 있습니다.

핸들러

핸들러는 명령 및 요청을 결과로 실제로 변환하는 함수입니다. 핸들러는 일반적으로 HTTP 요청을 전송합니다. 핸들러를 미들웨어와 함께 구성하여 동작을 강화할 수 있습니다. 핸들러는 `Aws\CommandInterface` 및 `Psr\Http\Message\RequestInterface`를 받아 `Aws\ResultInterface`와 함께 실행되거나 `Aws\Exception\AwsException` 이유와 함께 거부되는 promise를 반환하는 함수입니다.

다음은 각 호출에 대해 동일한 모의(mock) 결과를 반환하는 핸들러입니다.

```
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);
};
```

그런 다음 클라이언트 생성자에서 handler 옵션을 제공하여 SDK 클라이언트에서 이 핸들러를 사용할 수 있습니다.

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

setHandler의 Aws\ClientInterface 메서드를 사용하여 클라이언트의 핸들러를 생성한 이후에 핸들러를 변경할 수도 있습니다.

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

> **Note**
>
> Aws\MultiRegionClient의 useCustomHandler 메서드를 사용하여 클라이언트의 핸들러를 생성한 이후에 핸들러를 변경할 수도 있습니다.
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

모의(mock) 핸들러

SDK를 사용하는 테스트를 작성할 경우 MockHandler를 사용하는 것이 좋습니다. Aws\MockHandler를 사용하여 모의(mock) 결과를 반환하거나 모의(mock) 예외를 발생할 수 있습니다. 결과 또는 예외를 대기열에 넣으면 MockHandler가 FIFO 순으로 대기열에서 제거합니다.

```
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();
```

미들웨어

미들웨어는 특수 유형의 상위 수준 함수로서, 명령을 전송하는 동작을 강화하고 “다음” 핸들러에 위임합니다. 미들웨어 함수는 `Aws\CommandInterface` 및 `Psr\Http\Message\RequestInterface`를 받아 `Aws\ResultInterface`와 함께 실행되거나 `Aws\Exception\AwsException` 이유와 함께 거부되는 promise를 반환합니다.

미들웨어는 통과하는 명령, 요청 또는 결과를 수정하는 상위 수준 함수입니다. 미들웨어는 다음과 같은 형식으로 되어 있습니다.

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

미들웨어는 실행할 명령과 선택적 요청 객체를 수신합니다. 미들웨어는 요청 및 명령을 보강하거나 그대로 두도록 선택할 수 있습니다. 그런 후 체인 내 다음 핸들을 호출하거나 다음 핸들러를 단락시키고 promise를 반환하도록 선택할 수 있습니다. 다음 핸들러를 호출하여 생성되는 promise를 해당하는 `then` 메서드로 보강하여 이벤트 결과 또는 오류를 수정한 후 promise를 미들웨어의 스택에 다시 반환할 수 있습니다.

HandlerList

SDK는 `Aws\HandlerList`를 사용하여 명령을 실행할 때 사용되는 미들웨어와 핸들러를 관리합니다. 각 SDK 클라이언트에는 `HandlerList`가 있고, 이 `HandlerList`는 복제되어 클라이언트에서 생성되는 각 명령에 추가됩니다. 미들웨어와 기본 핸들러를 연결한 후 미들웨어를 클라이언트의 `HandlerList`에 추가하여 클라이언트에서 생성되는 각 명령에서 사용할 수 있습니다. 특정 명령이 소유한 `HandlerList`를 수정하여 해당 명령에서 미들웨어를 추가 및 제거할 수 있습니다.

`HandlerList`는 핸들러를 래핑하는 데 사용되는 미들웨어 스택을 나타냅니다. 미들웨어 목록과 핸들러를 래핑하는 순서를 관리할 수 있도록 `HandlerList`는 명령을 전송하는 수명 주기의 일부를 나타내는 명명된 단계로 미들웨어 스택을 분할합니다.

1. `init` - 기본 파라미터 추가
2. `validate` - 필수 파라미터 확인
3. `build` - 전송을 위한 HTTP 요청 직렬화
4. `sign` - 직렬화된 HTTP 요청에 서명
5. <handler>(단계가 아니지만 실제 전송을 수행함)

init

이 수명 주기 단계는 명령 초기화를 나타내며 요청이 아직 직렬화되지 않았습니다. 이 단계는 일반적으로 명령에 기본 파라미터를 추가하는 데 사용됩니다.

`init` 및 `appendInit` 메서드를 사용하여 `prependInit` 단계에 미들웨어를 추가할 수 있습니다. 여기서 `appendInit`은 `prepend` 목록의 끝에 미들웨어를 추가하고, `prependInit`은 `prepend` 목록의 앞에 미들웨어를 추가합니다.

```
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');
```

검증

이 수명 주기 단계는 명령의 입력 파라미터를 확인하는 데 사용됩니다.

`validate` 및 `appendValidate` 메서드를 사용하여 `prependValidate` 단계에 미들웨어를 추가할 수 있습니다. 여기서 `appendValidate`은 `validate` 목록의 끝에 미들웨어를 추가하고, `prependValidate`은 `validate` 목록의 앞에 미들웨어를 추가합니다.

```
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');
```

빌드

이 수명 주기 단계는 실행 중인 명령에 대한 HTTP 요청을 직렬화하는 데 사용됩니다. 다운스트림 수명 주기 이벤트는 명령과 PSR-7 HTTP 요청을 수신합니다.

build 및 appendBuild 메서드를 사용하여 prependBuild 단계에 미들웨어를 추가할 수 있습니다. 여기서 appendBuild은 build 목록의 끝에 미들웨어를 추가하고, prependBuild은 build 목록의 앞에 미들웨어를 추가합니다.

```
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

이 수명 주기 단계는 일반적으로 네트워크를 통해 전송하기 이전에 HTTP 요청에 서명하는 데 사용됩니다. 일반적으로 서명 오류를 방지하기 위해 서명한 이후에는 HTTP 요청을 변경하지 않는 것이 좋습니다.

핸들러에서 HTTP 요청을 전송하기 이전에 수행되는 HandlerList의 마지막 단계입니다.

sign 및 appendSign 메서드를 사용하여 prependSign 단계에 미들웨어를 추가할 수 있습니다. 여기서 appendSign은 sign 목록의 끝에 미들웨어를 추가하고, prependSign은 sign 목록의 앞에 미들웨어를 추가합니다.

```
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');
```

사용 가능한 미들웨어

SDK는 클라이언트의 동작을 보강하거나 명령의 실행을 관찰하는 데 사용할 수 있는 다양한 미들웨어를 제공합니다.

mapCommand

`Aws\Middleware::mapCommand` 미들웨어는 명령을 HTTP 요청으로 직렬화하기 전에 명령을 수정해야 하는 경우에 유용합니다. 예를 들어, `mapCommand`는 기본 파라미터를 확인하거나 추가하는 데 사용할 수 있습니다. `mapCommand` 함수는 `Aws\CommandInterface` 객체를 받아 `Aws\CommandInterface` 객체를 반환하는 collable을 받습니다.

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

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

// Apply a custom middleware named "add-param" to the "init" lifecycle step
$command->getHandlerList()->appendInit(
    Middleware::mapCommand(function (CommandInterface $command) {
        $command['Bucket'] = 'amzn-s3-demo-bucket';
        // Be sure to return the command!
        return $command;
    }),
    'add-param'
);
```

mapRequest

`Aws\Middleware::mapRequest` 미들웨어는 요청을 직렬화한 후 전송하기 이전에 수정해야 하는 경우에 유용합니다. 예를 들어, 요청에 사용자 지정 HTTP 헤더를 추가하는 데 사용할 수 있습니다. `mapRequest` 함수는 `Psr\Http\Message\RequestInterface` 인수를 받아 `Psr\Http\Message\RequestInterface` 객체를 반환하는 collable을 받습니다.

```
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'
);
```

이제 명령을 실행하면 명령이 사용자 지정 헤더와 함께 전송됩니다.

> **Important**
>
> 미들웨어는 `build` 단계를 마칠 때 핸들러 목록에 추가되었습니다. 따라서 이 미들웨어를 호출하기 이전에 요청이 생성된 것을 알 수 있습니다.

mapResult

`Aws\Middleware::mapResult` 미들웨어는 명령 실행 결과를 수정해야 하는 경우에 유용합니다. `mapResult` 함수는 `Aws\ResultInterface` 인수를 받아 `Aws\ResultInterface` 객체를 반환하는 collable을 받습니다.

```
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;
    })
);
```

이제 명령을 실행하면 반환되는 결과에 foo 속성이 포함되어 있습니다.

기록

history 미들웨어는 SDK에서 예상한 명령을 실행하고, 예상한 HTTP 요청을 전송하고, 예상한 결과를 수신했는지 테스트하는 데 유용합니다. 이 미들웨어는 웹 브라우저의 검색 기록과 비슷한 역할을 합니다.

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

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

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

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

Aws\History 내역 컨테이너에는 기본적으로 10개 항목이 저장됩니다. 이 수를 초과하면 항목이 제거됩니다. 유지할 항목 수를 생성자에 전달하여 항목 수를 사용자 지정할 수 있습니다.

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

history 미들웨어를 통과하는 요청을 실행한 후 내역 컨테이너를 검사할 수 있습니다.

```
// The object is countable, returning the number of entries in the container
count($history);

// The object is iterable, yielding each entry in the container
foreach ($history as $entry) {
    // You can access the command that was executed
    var_dump($entry['command']);
```

```
    // The request that was serialized and sent
    var_dump($entry['request']);
    // The result that was received (if successful)
    var_dump($entry['result']);
    // The exception that was received (if a failure occurred)
    var_dump($entry['exception']);
}

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

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

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

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

tap

`tap` 미들웨어는 관찰자로 사용됩니다. 미들웨어의 체인을 통해 명령을 전송할 경우 이 미들웨어를 사용하여 함수를 호출할 수 있습니다. `tap` 함수는 `Aws\CommandInterface` 및 실행 중인 `Psr\Http\Message\RequestInterface`(옵션)를 받는 collable 함수입니다.

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

사용자 지정 핸들러 생성

핸들러는 `Aws\CommandInterface` 객체와 `Psr\Http\Message\RequestInterface` 객체를 받아 `GuzzleHttp\Promise\PromiseInterface`와 함께 실행되거나 `Aws\ResultInterface`와 함께 거부되는 `Aws\Exception\AwsException`를 반환하는 단순한 함수입니다.

SDK에는 다양한 `@http` 옵션이 있지만 핸들러에서는 다음 옵션을 사용하는 방법만 알면 됩니다.

- connect_timeout
- debug
- decode_content(선택 사항)
- delay
- progress(선택 사항)
- proxy
- sink
- synchronous(선택 사항)
- stream(선택 사항)
- 제한 시간
- verify
- http_stats_receiver(선택 사항) - stats 구성 파라미터를 사용하여 요청된 경우 HTTP 전송 통계의 연결 배열을 사용하여 호출하는 함수

이 옵션을 선택 사항으로 지정한 경우 핸들러는 옵션을 처리하거나 거부된 promise를 반환할 수 있어야 합니다.

특정 `@http` 옵션 처리 외에도 핸들러는 다음과 같은 형식의 `User-Agent` 헤더를 추가해야 합니다. 여기서 “3.X”를 `Aws\Sdk::VERSION`으로 바꿀 수 있고 “HandlerSpecificData/version ...”을 사용 중인 핸들러별 User-Agent 문자열로 바꾸어야 합니다.

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

AWS SDK for PHP 버전 3의 스트림

PSR-7 HTTP 메시지 표준 통합의 일부로, AWS SDK for PHP는 PSR-7 StreamInterface를 내부적으로 PHP 스트림에 대한 추상화로 사용합니다. S3: :PutObject 명령의 Body 파라미터와 같이 입력 필드가 블롭으로 정의된 모든 명령은 문자열, PHP 스트림 리소스 또는 `Psr\Http\Message\StreamInterface` 인스턴스로 해결할 수 있습니다.

> **Warning**
>
> SDK는 명령에 입력 파라미터로 제공된 모든 원시 PHP 스트림 리소스의 소유권을 갖습니다. 스트림은 사용자를 대신하여 사용되고 닫힙니다.
> SDK 작업과 코드 간에 스트림을 공유해야 하는 경우 스트림을 명령 파라미터로 포함시키기 전에 `GuzzleHttp\Psr7\Stream`의 인스턴스 안에 래핑합니다. SDK는 스트림을 사용하므로, 코드는 스트림 내부 커서의 이동을 고려해야 합니다. Guzzle 스트림은 PHP의 가비지 수집기에서 삭제될 때 기본 스트림 리소스의 `fclose`를 호출하므로, 스트림을 직접 닫을 필요가 없습니다.

스트림 데코레이터

Guzzle은 SDK와 Guzzle이 명령에 입력 파라미터로 제공된 스트리밍 리소스와 상호 작용하는 방식을 제어하는 데 사용할 수 있는 여러 개의 스트림 데코레이터를 제공합니다. 이러한 데코레이터는 핸들러가 지정된 스트림에서 읽고 검색하는 방법을 수정할 수 있습니다. 다음은 부분적인 목록입니다. 자세한 내용은 GuzzleHttpPsr7 리포지토리에서 확인할 수 있습니다.

AppendStream

GuzzleHttp\Psr7\AppendStream

여러 스트림에서 하나씩 차례로 읽습니다.

```
use GuzzleHttp\Psr7;

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

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

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

CachingStream

GuzzleHttp\Psr7\CachingStream

검색할 수 없는 스트림에서 이전에 읽은 바이트에 대한 검색을 허용하기 위해 사용됩니다. 이 기능은 스트림을 되감아야 하기 때문에(예를 들어 리디렉션의 결과로) 검색할 수 없는 개체 본문 전송에 실패할 때 유용합니다. 이전에 읽은 바이트가 먼저 메모리에 캐시된 다음 디스크에 캐시되도록 원격 스트림에서 읽은 데이터는 PHP 임시 스트림에서 버퍼링됩니다.

```
use GuzzleHttp\Psr7;

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

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

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

InflateStream

GuzzleHttp\Psr7\InflateStream

PHP의 zlib.inflate 필터를 사용하여 gzip으로 압축된 콘텐츠를 inflate 또는 deflate합니다.

이 스트림 데코레이터는 지정된 스트림의 처음 10바이트를 건너뛰어서 gzip 헤더를 제거하고, 제공된 스트림을 PHP 스트림 리소스로 변환한 다음, zlib.inflate 필터를 추가합니다. 그런 다음 스트림은 Guzzle 스트림으로 사용될 Guzzle 스트림 리소스로 다시 변환됩니다.

LazyOpenStream

GuzzleHttp\Psr7\LazyOpenStream

스트림에서 I/O 작업이 수행된 후에만 열리는 파일에 지연 읽기 또는 쓰기를 수행합니다.

```
use GuzzleHttp\Psr7;

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

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

LimitStream

GuzzleHttp\Psr7\LimitStream

기존 스트림 객체의 하위 집합이나 조각을 읽는 데 사용됩니다. 이 기능은 큰 파일을 청크로 전송될 작은 부분으로 분리하는 경우에 유용합니다(예: Amazon S3 멀티파트 업로드 API).

```
use GuzzleHttp\Psr7;

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

// Limit the size of the body to 1024 bytes and start reading from byte 2048
$stream = new Psr7\LimitStream($original, 1024, 2048);
echo $stream->getSize();
// >>> 1024
echo $stream->tell();
// >>> 0
```

NoSeekStream

GuzzleHttp\Psr7\NoSeekStream

스트림을 래핑하고 검색을 허용하지 않습니다.

```
use GuzzleHttp\Psr7;

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

echo $noSeek->read(3);
// foo
```

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

PumpStream

GuzzleHttp\Psr7\PumpStream

PHP collable에서 데이터를 가져오는 읽기 전용 스트림을 제공합니다.

제공된 collable을 호출하면 PumpStream은 읽기 요청된 만큼의 데이터를 collable에 전달합니다. collable은 이 값을 무시하고 요청된 것보다 더 적거나 더 많은 바이트를 반환하도록 선택할 수 있습니다. 제공된 collable에서 반환하는 추가 데이터는 PumpStream의 read() 함수를 사용하여 드레이닝될 때까지 내부적으로 버퍼링됩니다. 읽을 데이터가 더 이상 없을 경우 제공된 collable은 false를 반환해야 합니다.

스트림 데코레이터 구현

스트림 데코레이터 생성은 GuzzleHttp\Psr7\StreamDecoratorTrait 덕분에 매우 쉽습니다. 이 특성은 기본 스트림에 프록시하여 `Psr\Http\Message\StreamInterface`를 구현하는 메서드를 제공합니다. `use`를 `StreamDecoratorTrait`하고 사용자 지정 메서드를 구현합니다.

예를 들어 스트림에서 마지막 바이트를 읽을 때마다 특정 함수를 호출하려 한다고 가정합니다. `read()` 메서드를 재정의하여 이 작업을 구현할 수 있습니다.

```
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;
    }
}
```

이상과 같이 이 데코레이터를 기존 스트림에 추가하고 사용할 수 있습니다.

```
use GuzzleHttp\Psr7;

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

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

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

AWS SDK for PHP 버전 3에서 페이지 매김된 결과 사용

일부 AWS 서비스 작업은 페이지 지정되며 잘린 결과로 응답합니다. 예를 들어, Amazon S3`ListObjects` 작업은 한 번에 최대 1,000개의 객체만 반환합니다. 이와 같은 작업(일반적으로 “list” 또는 “describe”라는 접두사가 붙음)을 수행하려면 토큰(또는 마커) 파라미터와 함께 후속 요청을 수행하여 전체 결과 집합을 검색해야 합니다.

페이지네이터는 개발자가 페이지 매김된 API를 쉽게 사용할 수 있도록 이 프로세스에서 추상화 역할을 수행하는 AWS SDK for PHP의 기능입니다. 페이지네이터는 본질적으로 결과의 반복자입니다. 이 기능은 클라이언트의 `getPaginator()` 메서드를 통해 생성됩니다. `getPaginator()`를 호출할 때 작

업 이름과 작업 인수를 제공해야 합니다(작업을 실행할 때 수행한 것과 동일한 방법 사용). foreach를 사용하는 페이지네이터 객체를 반복하여 개별 Aws\Result 객체를 가져올 수 있습니다.

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

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

페이지네이터 객체

getPaginator() 메서드에서 반환된 객체는 Aws\ResultPaginator 클래스의 인스턴스입니다. 이 클래스는 PHP의 고유 iterator 인터페이스를 구현하며, 이러한 이유로 foreach와 함께 작동합니다. iterator_to_array와 같은 iterator 함수와도 함께 사용할 수 있으며 LimitIterator 객체와 같은 SPL iterator와 잘 통합됩니다.

페이지네이터 객체는 한 번에 한 "페이지"의 결과만 보유하며 지연 실행됩니다. 따라서 결과의 동시 페이지를 산출하는 데 필요한 수의 요청만 수행합니다. 예를 들어, Amazon S3ListObjects 작업은 한 번에 최대 1,000개의 객체만 반환하므로, 버킷에 ~10,000개의 객체가 있는 경우 페이지네이터는 총 10개의 요청을 수행해야 합니다. 결과를 반복하면 반복을 시작할 때 첫 번째 요청이 실행되고 루프의 두 번째 반복에서 두 번째 요청이 실행되며 같은 방식으로 계속됩니다.

결과의 데이터 열거

페이지네이터 객체에는 search()라는 메서드가 있습니다. 이 메서드를 사용하여 결과 집합 내의 데이터에 대한 반복자를 생성할 수 있습니다. search()를 호출할 때 JMESPath 표현식을 제공하여 어떤 데이터를 추출할지를 지정합니다. search()를 호출하면 결과의 각 페이지에서 표현식의 결과를 산출하는 반복자가 반환됩니다. 반환된 반복자를 반복하므로 이 값은 늦게 평가됩니다.

다음 예제는 이전 코드 예제와 동등하지만, 더 간결하게 하기 위해 ResultPaginator::search() 메서드를 사용합니다.

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

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

JMESPath 표현식을 사용하면 상당히 복잡한 작업을 수행할 수 있습니다. 예를 들어, 모든 객체 키와 공통 접두사를 인쇄하려는 경우(예: 버킷의 `ls` 실행) 다음을 수행할 수 있습니다.

```
// 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";
}
```

비동기 페이지 매김

`each()`의 `Aws\ResultPaginator` 메서드에 콜백을 제공하여 페이지네이터의 결과를 비동기적으로 반복할 수 있습니다. 페이지네이터에서 산출되는 각 값에 대해 콜백이 호출됩니다.

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

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

> **Note**
>
> `each()` 메서드를 사용하면 다른 요청을 비동기적으로 동시에 전송하면서 API 작업의 결과에 페이지를 매길 수 있습니다.

기본 코루틴 기반 promise에서 콜백의 null이 아닌 반환 값이 산출됩니다. 따라서 남은 항목을 계속 반복하여 본질적으로 다른 promise를 반복에 병합하기 전에 해결해야 하는 콜백의 promise를 반환할 수 있습니다. 콜백에서 반환되는 마지막 null이 아닌 값은 다운스트림 promise까지 promise를 이행하는 결

과입니다. 마지막 반환 값이 promise인 경우 해당 promise의 해결은 다운스트림 promise를 이행하거나 거부하는 결과입니다.

```
// Delete all keys that end with "Foo"
$promise = $results->each(function ($result) use ($s3Client) {
    if (substr($result['Key'], -3) === 'Foo') {
        // Merge this promise into the iterator
        return $s3Client->deleteAsync([
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'Foo'
        ]);
    }
});

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

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

AWS SDK for PHP 버전 3의 웨이터

Waiter를 사용하면 리소스를 폴링하여 리소스가 특정 상태로 전환될 때까지 대기할 수 있는 추상화된 방법을 제공하여 시스템을 더 쉽고 일관되게 사용할 수 있습니다. 클라이언트에서 지원되는 Waiter 목록은 서비스 클라이언트의 API 설명서에서 확인할 수 있습니다. 그곳으로 이동하려면 API 설명서의 클라이언트 페이지로 이동하여 특정 버전 번호 (날짜로 표시) 로 이동한 다음 'Waiters' 섹션으로 스크롤합니다. 이 링크를 클릭하면 S3의 웨이터 섹션으로 이동합니다.

다음 예에서 Amazon S3 클라이언트는 버킷을 생성하는 데 사용됩니다. Waiter는 버킷이 존재할 때까지 대기하는 데 사용됩니다.

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

waiter에서 버킷을 지나치게 많이 폴링해야 하는 경우 \RuntimeException 예외를 발생합니다.

Waiter 구성

Waiter는 구성 옵션의 결합형 배열을 기반으로 합니다. 특정 waiter에서 사용되는 모든 옵션은 기본값이 있지만, 다른 대기 전략을 지원하도록 재정의할 수 있습니다.

@waiter 옵션의 결합형 배열을 클라이언트의 $args 및 waitUntil() 메서드의 getWaiter() 인수에 전달하여 waiter 구성 옵션을 수정할 수 있습니다.

```
// Providing custom waiter configuration options to a waiter
$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'amzn-s3-demo-bucket',
    '@waiter' => [
        'delay'       => 3,
        'maxAttempts' => 10
    ]
]);
```

delay (int)

폴링 시도 사이의 지연 시간(초)입니다. 각 waiter에는 기본 delay 구성 값이 있지만, 특정 사용 사례에 대해 이 설정을 수정해야 할 수 있습니다.

maxAttempts (int)

waiter를 실패로 처리하기 이전의 최대 폴링 시도 횟수입니다. 이 옵션은 리소스를 무기한으로 대기하지 않도록 해줍니다. 각 waiter에는 기본 maxAttempts 구성 값이 있지만, 특정 사용 사례에 대해 이 설정을 수정해야 할 수 있습니다.

initDelay (int)

최초 폴링 시도 이전에 대기할 시간(초)입니다. 이 옵션은 원하는 상태로 전환되는 데 시간이 걸릴 것을 알고 있는 리소스를 대기할 때 유용합니다.

before (callable)

각 시도 전에 호출되는 PHP callable 함수입니다. callable 함수는 실행할 Aws\CommandInterface 명령과 지금까지 실행된 횟수를 기준으로 호출됩니다. before callable 함수를 사용하여 실행되기 전에 명령을 수정하거나 진행률 정보를 제공할 수 있습니다.

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

비동기적 대기

비동기적으로 대기하는 이외에 waiter를 호출하여 다른 요청을 보내거나 한 번에 여러 리소스를 대기하면서 비동기적으로 대기할 수 있습니다.

클라이언트의 `getWaiter($name, array $args = [])` 메서드를 사용하여 클라이언트에서 waiter를 검색하여 waiter promise에 액세스할 수 있습니다. waiter의 `promise()` 메서드를 사용하여 waiter를 시작합니다. waiter promise는 waiter에서 실행된 마지막 `Aws\CommandInterface`를 통해 이행되며, 오류 시 `RuntimeException`과 함께 거부됩니다.

```
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();
```

일부 강력하고 상대적으로 낮은 오버헤드 사용 사례에서 promise 기반 waiter API를 노출할 수 있습니다. 예를 들어, 여러 리소스를 대기하고, 확인된 첫 번째 waiter를 처리하려는 경우 어떻게 될까요?

```
use Aws\CommandInterface;

// Create an array of waiter promises
$promises = [
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'a'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'b'])->promise(),
    $s3Client->getWaiter('BucketExists', ['Bucket' => 'c'])->promise()
];

// Initiate a race between the waiters, fulfilling the promise with the
// first waiter to complete (or the first bucket to become available)
$any = Promise\any($promises)
    ->then(function (CommandInterface $command) {
        // This is invoked with the command that succeeded in polling the
        // resource. Here we can know which bucket won the race.
        echo "The {$command['Bucket']} waiter completed first!\n";
    });

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

AWS SDK for PHP 버전 3의 JMEsPath 표현식

JMESPath를 사용하여 JSON 문서에서 요소를 추출하는 방법을 선언적으로 지정할 수 있습니다. AWS SDK for PHP에서는 jmespath.php에 의존하여 일부 상위 수준의 추상화(예: AWS SDK for PHP 버전 3의 페이지네이터, AWS SDK for PHP 버전 3의 Waiter)를 제공하고, `Aws\ResultInterface` 및 `Aws\ResultPaginator`에서 JMESPath 검색을 노출합니다.

온라인 JMESPath 예제를 통해 브라우저에서 JMESPath를 사용해 볼 수 있습니다. JMESPath 사양에서 언어(사용 가능한 표현식 및 함수 포함)에 대해 자세히 알아볼 수 있습니다.

AWS CLI는 JMESPath를 지원합니다. CLI 출력을 위해 작성된 표현식은 AWS SDK for PHP용으로 작성된 표현식과 100% 호환됩니다.

결과에서 데이터 추출

`Aws\ResultInterface` 인터페이스에는 JMESPath 표현식을 기반으로 결과 모델에서 데이터를 추출하는 `search($expression)` 메서드가 있습니다. JMESPath 표현식을 사용하여 결과 객체에서 데이터를 쿼리하면 보일러플레이트 조건부 코드를 제거하고, 추출 중인 데이터를 자세히 나타낼 수 있습니다.

작동 방식을 보여 주기 위해 먼저 아래와 같은 기본 JSON 출력으로 시작합니다. 여기서는 별도의 EC2 인스턴스에 연결된 두 개의 Amazon Elastic Block Store (Amazon EBS) 볼륨을 설명합니다.

```
$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"
        }
    }
}
```

먼저 다음 명령을 사용하여 볼륨 목록의 첫 번째 볼륨만 검색할 수 있습니다.

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

이제 `wildcard-index` expression `[*]`를 사용하여 전체 목록을 반복하고 세 요소를 추출한 후 `VolumeId`는 `ID`로, `AvailabilityZone`은 `AZ`로 이름을 바꾸고 `Size`는 `Size`로 그대로 둡니다. 이러한 요소를 추출한 후 `multi-hash` 표현식을 `wildcard-index` 표현식 뒤에 사용하여 이름을 바꿉니다.

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

그러면 다음과 같은 PHP 데이터 배열이 제공됩니다.

```
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)
  }
}
```

또한 multi-hash 표기법에서는 key1.key2[0].key3과 같은 체인 키를 사용하여 구조 내에 깊이 중첩된 요소를 추출할 수 있습니다. 다음 예에서는 간단히 Attachments[0].InstanceId라는 별칭이 지정된 InstanceId 키를 사용하여 이 작업을 보여 줍니다. 대부분의 경우 JMESPath 표현식에서는 공백을 무시합니다.

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

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

이전 표현식은 다음 데이터를 출력합니다.

```
array(2) {
  [0] =>
  array(4) {
    'ID' =>
    string(12) "vol-e11a5288"
    'InstanceId' =>
```

```
    string(10) "i-a071c394"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(30)
  }
  [1] =>
  array(4) {
    'ID' =>
    string(12) "vol-2e410a47"
    'InstanceId' =>
    string(10) "i-4b41a37c"
    'AZ' =>
    string(10) "us-west-2a"
    'Size' =>
    int(8)
  }
}
```

multi-list expression:[key1, key2]를 사용하여 여러 요소를 필터링할 수도 있습니다. 이렇게 하면 유형과 상관없이 필터링된 모든 속성이 객체당 단 하나의 순서가 지정된 목록으로 서식 지정됩니다.

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

이전 검색을 실행하면 다음과 같은 데이터가 생성됩니다.

```
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)
  }
}
```

특정 필드 값을 기준으로 결과를 필터링하려면 `filter` 표현식을 사용합니다. 다음 예제 쿼리는 `us-west-2a` 가용 영역의 볼륨만 출력합니다.

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

JMESPath에서는 함수 표현식도 지원합니다. 위와 동일한 쿼리를 실행하지만 이번에는 AWS 리전에서 "us-"로 시작하는 모든 볼륨을 검색한다고 가정합니다. 다음 표현식에서는 `starts_with` 문자열 리터럴을 전달하여 `us-` 함수를 사용합니다. 그런 다음 필터 투영을 통해 `true`를 반환한 필터 조건자의 결과만 전달하여 이 함수의 결과를 `true` JSON 리터럴 값과 비교합니다.

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

페이지네이터에서 데이터 추출

AWS SDK for PHP 버전 3의 페이지네이터 가이드에서 살펴본 대로 `Aws\ResultPaginator` 객체는 페이징 가능한 API 작업에서 결과를 출력하는 데 사용됩니다. AWS SDK for PHP를 사용하면 `Aws\ResultPaginator` 객체에서 필터링된 데이터를 추출하여 반복할 수 있습니다. 이때 JMESPath 표현식의 결과가 맵 함수인 반복자에 대해 flat-map을 구현해야 합니다.

버킷에서 1MB보다 큰 객체만 출력하는 `iterator`를 생성한다고 가정합니다. 이렇게 하려면 `ListObjects` 페이지네이터를 생성한 다음 `search()` 함수를 페이지네이터에 적용하여 페이지 지정된 데이터에 대해 flat-map 반복자를 생성합니다.

```
$result = $s3Client->getPaginator('ListObjects', ['Bucket' => 't1234']);
$filtered = $result->search('Contents[?Size > `1048576`]');

// The result yielded as $data will be each individual match from
// Contents in which the Size attribute is > 1048576
```

```
foreach ($filtered as $data) {
    var_dump($data);
}
```

AWS SDK for PHP 버전 3에서 AWS 서비스 호출

다음 섹션에는 AWS SDK for PHP를 사용하여 AWS 서비스를 사용하는 방법을 보여주는 예제, 자습서, 작업 및 가이드가 포함되어 있습니다.

주제

- AWS SDK for PHP 버전 3의 기능 및 옵션 사용
- AWS SDK for PHP에 대한 지침이 포함된 코드 예제

AWS SDK for PHP 버전 3의 기능 및 옵션 사용

AWS SDK for PHP 버전 3에서는 AWS 서비스 APIs 작업을 위한 추가 기능 및 옵션에 대한 지원을 제공합니다. 이 항목의 섹션에서는 이러한 옵션을 서비스별로 다룹니다.

주제

- AWS SDK for PHP 버전 3에서 DynamoDB 세션 핸들러 사용
- AWS SDK for PHP 버전 3의 Amazon S3 기능 및 옵션

AWS SDK for PHP 버전 3에서 DynamoDB 세션 핸들러 사용

DynamoDB 세션 핸들러는 개발자가 Amazon DynamoDB를 세션 저장소로 사용할 수 있는 PHP용 사용자 지정 세션 핸들러입니다. DynamoDB를 세션 스토리지로 사용하면 세션을 로컬 파일 시스템에서 공유 위치로 이동하여 분산 웹 애플리케이션에서 세션 처리 중에 발생하는 문제를 완화할 수 있습니다. DynamoDB는 빠르고 확장 가능하며 설정이 쉬우며 데이터 복제를 자동으로 처리합니다.

DynamoDB 세션 핸들러는 진정한 드롭인 교체가 가능하도록 `session_set_save_handler()` 함수를 사용하여 DynamoDB 작업을 PHP의 고유 세션 함수에 후크합니다. 여기에는 PHP의 기본 세션 핸들러의 일부인 세션 잠금 및 가비지 수집과 같은 기능에 대한 지원이 포함됩니다.

DynamoDB 서비스에 대한 자세한 내용은 Amazon DynamoDB. 홈 페이지를 참조하세요.

기본 사용법

1단계: 핸들러 등록

먼저 세션 핸들러를 인스턴스화하고 등록합니다.

```
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();
```

2단계. 세션을 저장할 테이블 생성

실제로 세션 핸들러를 사용하기 전에 먼저 세션을 저장할 테이블을 생성해야 합니다. Amazon DynamoDB용AWS 콘솔을 사용하거나 AWS SDK for PHP를 사용하여 이 작업을 사전에 수행할 수 있습니다.

이 테이블을 생성할 때는 'id'를 기본 키 이름으로 사용합니다. 또한 세션의 자동 가비지 수집을 이용하려면 'expires' 속성을 사용해 Time To Live 속성을 설정하는 것이 좋습니다.

3단계. 일반적인 방식으로 PHP 세션 사용

세션 핸들러가 등록되고 테이블이 생성된 후에는 일반적으로 PHP의 기본 세션 핸들러를 사용할 때와 똑같은 방식으로 $_SESSION superglobal을 사용하여 세션에서 쓰고 읽을 수 있습니다. DynamoDB 세션 핸들러는 DynamoDB와의 상호 작용을 캡슐화하고 추상화하며 이 핸들러를 통해 PHP의 고유 세션 함수와 인터페이스를 쉽게 사용할 수 있습니다.

```
// 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();
```

구성

다음 옵션을 사용하여 세션 핸들러의 동작을 구성할 수 있습니다. 모든 옵션은 선택 사항이지만, 기본값이 무엇인지 이해해야 합니다.

table_name

세션을 저장할 DynamoDB 테이블의 이름입니다. 기본값은 `'sessions'`입니다.

hash_key

DynamoDB 세션 테이블에 있는 해시 키의 이름입니다. 기본값은 `'id'`입니다.

data_attribute

세션 데이터가 저장된 DynamoDB 세션 테이블에 있는 속성의 이름입니다. 기본값은 `'data'`입니다.

data_attribute_type

세션 데이터가 저장된 DynamoDB 세션 테이블에 있는 속성의 유형입니다. 기본값은 `'string'`이지만, 선택에 따라 `'binary'`으로 설정할 수 있습니다.

session_lifetime

가비지 수집되기 전 비활성 세션의 수명입니다. 이 옵션을 제공하지 않으면 사용되는 실제 수명 값은 `ini_get('session.gc_maxlifetime')`입니다.

session_lifetime_attribute

세션 만료 시간이 저장된 DynamoDB 세션 테이블에 있는 속성의 이름입니다. 기본값은 `'expires'`입니다.

consistent_read

세션 핸들러가 `GetItem` 작업에 일관된 읽기를 사용하는지 여부입니다. 기본값은 `true`입니다.

locking

세션 잠금을 사용할지 여부입니다. 기본값은 `false`입니다.

batch_config

가비지 수집 중에 배치 삭제하는 데 사용되는 구성입니다. 이 옵션은 DynamoDB WriteRequestBatch 객체에 직접 전달됩니다. `SessionHandler::garbageCollect()`를 통해 가비지 수집을 수동으로 트리거합니다.

max_lock_wait_time

세션 핸들러가 포기하기 전에 잠금을 획득하기 위해 대기해야 하는 최대 시간(초)입니다. 기본값은 `10`이며 세션 잠금에만 사용됩니다.

min_lock_retry_microtime

세션 핸들러가 잠금을 획득하기 위한 시도 간에 대기해야 하는 최소 시간(마이크로초)입니다. 기본값은 10000이며 세션 잠금에만 사용됩니다.

max_lock_retry_microtime

세션 핸들러가 잠금을 획득하기 위한 시도 간에 대기해야 하는 최대 시간(마이크로초)입니다. 기본값은 50000이며 세션 잠금에만 사용됩니다.

세션 핸들러를 구성하려면 핸들러를 인스턴스화할 때 구성 옵션을 지정합니다. 다음 코드는 모든 구성 옵션이 지정된 예제입니다.

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

가격 책정

데이터 스토리지 및 데이터 전송 요금과 별도로, 테이블의 프로비저닝된 처리 능력을 기반으로 DynamoDB 사용과 관련된 비용이 계산됩니다(Amazon DynamoDB 요금 내역 참조). 처리량은 쓰기 용량 및 읽기 용량 단위로 측정됩니다. Amazon DynamoDB 홈페이지에는 다음과 같이 나와 있습니다.

읽기 용량 단위는 4KB 크기의 항목에 대해 초당 하나의 강력히 일관된 읽기(또는 초당 두 개의 최종적 일관된 읽기)를 나타냅니다. 쓰기 용량 단위는 1KB 크기의 항목에 대해 초당 하나의 쓰기를 나타냅니다.

궁극적으로 세션 테이블에 필요한 처리량과 비용은 예상 트래픽 및 세션 크기와 관련됩니다. 다음 표에서는 각 세션 함수에 대해 DynamoDB 테이블에서 수행되는 읽기 및 쓰기 작업의 양을 설명합니다.

`session_start()` 를 통한 읽기	• 1개의 읽기 작업입니다(`consisten t_read` 가 `false`인 경우 0.5만). • (조건부) 세션이 만료된 경우 세션을 삭제하기 위한 1개의 쓰기 작업입니다.
`session_start()` 를 통한 읽기(세션 잠금 사용)	• 최소 1개의 쓰기 작업입니다. • (조건부) 세션에서 잠금 획득 시 각 시도에 대해 추가 쓰기 작업입니다. 구성된 잠금 대기 시간 및 재시도 옵션을 기반으로 합니다. • (조건부) 세션이 만료된 경우 세션을 삭제하기 위한 1개의 쓰기 작업입니다.
`session_write_close()` 를 통한 쓰기	• 1개의 쓰기 작업입니다.
`session_destroy()` 를 통한 삭제	• 1개의 쓰기 작업입니다.
가비지 수집	• 만료된 세션에 대해 스캔할 테이블의 데이터 4KB당 0.5개의 읽기 작업입니다. • 삭제할 만료된 항목당 1개의 쓰기 작업입니다.

세션 잠금

DynamoDB 세션 핸들러는 PHP의 기본 세션 핸들러 동작을 모방하는 수동적 세션 잠금을 지원합니다. 특히 애플리케이션이 Ajax 요청이나 iframe을 사용하는 DynamoDB 세션에 액세스하는 경우 이 기능으로 인해 성능 병목 현상이 발생하고 비용이 급상승할 수 있기 때문에, 기본적으로 DynamoDB 세션 핸들러에서는 이 기능이 꺼져 있습니다. 이 기능을 활성화하기 전에 애플리케이션에 세션 잠금이 필요한지 여부를 신중하게 고려하세요.

세션 잠금을 활성화하려면 `'locking'`를 인스턴스화할 때 `true` 옵션을 `SessionHandler`로 설정합니다.

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

가비지 수집

'expires' 속성을 사용해 DynamoDB 테이블에 TTL 속성을 설정합니다. 그러면 세션에서 가비지를 자동으로 수집하기 때문에 직접 가비지를 수집할 필요 없습니다.

또한 DynamoDB 세션 핸들러는 일련의 `Scan` 및 `BatchWriteItem` 작업을 사용하여 세션 가비지 수집을 지원합니다. `Scan` 작업이 작동하는 방식의 특성 때문에 그리고 모든 만료된 세션을 찾아서 삭제하기 위해 가비지 수집 프로세스에는 많은 프로비저닝된 처리량이 필요합니다.

따라서 이를 통해 자동 가비지 수거를 지원하지 않습니다. 더 좋은 방법은 사용되는 처리량의 급증으로 인해 나머지 애플리케이션이 중단되지 않도록 피크가 아닌 시간에 가비지 수집 실행을 예약하는 것입니다. 예를 들어, 가비지 수집을 실행할 야간 cron 작업 트리거 스크립트가 있을 수 있습니다. 이 스크립트를 실행하려면 다음과 같은 작업을 수행해야 합니다.

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

$sessionHandler->garbageCollect();
```

또한 `'before'` 내에서 `'batch_config'` 옵션을 사용하여 가비지 수입 프로세스에서 수행되는 `BatchWriteItem` 작업에 지연을 도입할 수 있습니다. 이렇게 하면 가비지 수집이 완료되는 데 걸리는 시간의 양이 증가하지만, DynamoDB 세션 핸들러에서 수행되는 요청을 분산하여 가비지 수집 중에 프로비저닝된 처리 능력에 근접하거나 범위 이내로 처리량을 유지할 수 있습니다.

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

$sessionHandler->garbageCollect();
```

모범 사례

1. 애플리케이션 서버와 지리적으로 가장 가까운 AWS 리전 또는 동일한 리전에 세션 테이블을 생성합니다. 이렇게 하면 애플리케이션과 DynamoDB 데이터베이스 간에 지연 시간을 최소화할 수 있습니다.
2. 세션 테이블의 프로비저닝된 처리 능력을 신중하게 선택합니다. 애플리케이션의 예상 트래픽과 세션의 예상 크기를 고려합니다. 또는 테이블에 '온디맨드' 읽기/쓰기 용량 모드를 사용합니다.
3. AWS Management Console 또는 Amazon CloudWatch를 통해 사용된 처리량을 모니터링하고 애플리케이션의 요구 사항에 맞게 필요에 따라 처리량 설정을 조정합니다.
4. 세션 크기를 작게 유지합니다(1KB 미만이 이상적). 작은 세션이 더 좋은 성능을 나타내며 필요한 프로비저닝된 처리 능력이 더 적습니다.
5. 애플리케이션에 필요하지 않은 한 세션 잠금을 사용하지 마세요.
6. PHP의 내장 세션 가비지 수집 트리거를 사용하는 대신, cron 작업 또는 다른 예약 메커니즘을 통해 피크가 아닌 시간 중에 실행되도록 가비지 수집을 예약합니다. `'batch_config'` 옵션을 유리하게 사용합니다.

필수 IAM 권한

DynamoDB 세션 핸들러를 사용하려면 구성된 자격 증명에 이전 단계에서 생성한 DynamoDB 테이블을 사용할 권한이 있어야 합니다. 다음 IAM 정책에는 필요한 최소 권한이 포함되어 있습니다. 이 정책을 사용하려면 리소스 값을 이전에 생성한 테이블의 Amazon Resource Name (ARN)으로 바꿉니다. IAM 정책 생성 및 첨부에 대한 자세한 내용은 IAM 사용 설명서의 관리형 IAM 정책을 참조하세요.

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"
        }
    ]
}
```

AWS SDK for PHP 버전 3의 Amazon S3 기능 및 옵션

이 주제에서는 Amazon S3와 함께 작동하기 위해 AWS SDK for PHP 버전 3에서 제공하는 추가 기능 및 옵션에 대해 설명합니다.

주제

- AWS SDK for PHP 버전 3의 Amazon S3 다중 리전 클라이언트
- AWS SDK for PHP 버전 3의 Amazon S3 스트림 래퍼
- 파일 및 디렉터리 전송
- AWS SDK for PHP 버전 3의 Amazon S3 클라이언트 측 암호화
- 체크섬을 통한 데이터 무결성 보호

AWS SDK for PHP 버전 3의 Amazon S3 다중 리전 클라이언트

AWS SDK for PHP 버전 3은 모든 서비스와 함께 사용할 수 있는 일반 다중 리전 클라이언트를 제공합니다. 이를 통해 사용자는 명령에 `@region` 입력 파라미터를 제공하여 명령을 보낼 AWS 리전을 지정할 수 있습니다. 또한 SDK는 특정 Amazon S3 오류에 지능적으로 응답하고 그에 따라 명령을 다시 라우팅하는 Amazon S3용 다중 리전 클라이언트를 제공합니다. 이 기능을 통해 사용자는 동일한 클라이언트를 사용하여 다중 리전과 통신할 수 있습니다. 이는 버킷이 여러 리전에 있는 AWS SDK for PHP 버전 3의 Amazon S3 스트림 래퍼 사용자에게 특히 유용한 기능입니다.

기본 사용법

Amazon S3 클라이언트의 기본 사용 패턴은 표준 S3 클라이언트를 사용하든 다중 리전 클라이언트를 사용하든 상관없이 동일합니다. 명령 수준에서 유일한 사용량 차이는 `@region` 입력 파라미터를 사용하여 AWS 리전을 지정할 수 있다는 것입니다.

```
// 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

다중 리전 Amazon S3 클라이언트를 사용하면 영구적 리디렉션 예외가 발생하지 않습니다. 명령이 잘못된 리전에 전송되면 표준 Amazon S3 클라이언트는 `Aws\S3\Exception\PermanentRedirectException`의 인스턴스를 발생시킵니다. 다중 리전 클라이언트는 그 대신 명령을 올바른 리전으로 다시 디스패치합니다.

버킷 리전 캐시

Amazon S3 다중 리전 클라이언트는 지정된 버킷이 있는 AWS 리전의 내부 캐시를 유지합니다. 기본적으로 각 클라이언트에는 고유의 인 메모리 캐시가 있습니다. 클라이언트 또는 프로세스 간에 캐시를 공유하려면 `Aws\CacheInterface`의 인스턴스를 `bucket_region_cache` 옵션으로 다중 리전 클라이언트에 제공합니다.

```
use Aws\DoctrineCacheAdapter;
use Aws\Sdk;
use Doctrine\Common\Cache\ApcuCache;

$sdk = new Aws\Sdk([
    'version' => 'latest',
    'region' => 'us-west-2',
    'S3' => [
```

```
        'bucket_region_cache' => new DoctrineCacheAdapter(new ApcuCache),
    ],
]);
```

AWS SDK for PHP 버전 3의 Amazon S3 스트림 래퍼

Amazon S3 스트림 래퍼를 통해 `file_get_contents`, `fopen`, `copy`, `rename`, `unlink`, `mkdir` 및 `rmdir` 등과 같은 기본 제공 PHP 함수를 사용하여 Amazon S3에서 데이터를 저장하고 검색할 수 있습니다.

Amazon S3 스트림 래퍼를 사용하려면 등록해야 합니다.

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

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

그러면 `s3://` 프로토콜을 사용하여 Amazon S3에 저장된 버킷과 객체에 액세스할 수 있습니다. Amazon S3 스트림 래퍼는 버킷 이름 뒤에 슬래시와 선택적 객체 키 또는 `s3://<bucket>[/<key-or-prefix>]` 접두사가 오는 문자열(예:)을 허용합니다.

> **Note**
>
> 스트림 래퍼는 읽기 이상의 권한이 있는 객체 및 버킷으로 작업하도록 설계되었습니다. 즉, 사용자가 상호 작용해야 하는 버킷에 대해 `ListBucket`을 실행하고 객체에 대해 `GetObject`를 실행할 수 있는 권한이 있어야 합니다. 이 권한이 없는 경우 Amazon S3 클라이언트 작업을 직접 사용하는 것이 좋습니다.

데이터 다운로드

`file_get_contents`를 사용하여 객체의 콘텐츠를 선택할 수 있습니다. 하지만 이 함수는 객체의 전체 콘텐츠를 메모리로 로드하므로 주의하십시오.

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

더 큰 파일로 작업하거나 Amazon S3에서 데이터를 스트리밍해야 하는 경우 `fopen()`을 사용합니다.

```
// 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**
>
> 파일 쓰기 오류는 `fflush`를 호출하는 경우에만 반환되고, 플러시되지 않은 `fclose`를 호출하는 경우에는 반환되지 않습니다. 내부 `fclose`에 대한 응답으로 반환되는 오류에 상관없이 스트림을 닫을 경우 `true`에 대한 반환 값은 `fflush`입니다. PHP에서 구현되는 방법으로 인해 `file_put_contents`를 호출하는 경우에도 이 오류가 반환되지 않습니다.

검색 가능한 스트림 열기

"r" 모드에서 열린 스트림에서는 기본적으로 데이터를 읽을 수만 있고 검색할 수 없습니다. 이렇게 하면 Amazon S3에서 스트리밍 방식으로 데이터를 다운로드할 수 있으므로 이전에 읽은 바이트를 메모리로 버퍼링할 필요가 없습니다. 스트림을 검색할 수 있어야 하는 경우 `seekable`을 함수의 스트림 콘텍스트 옵션으로 전달합니다.

```
$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);
}
```

검색 가능한 스트림을 열어서 이전에 읽은 바이트를 검색할 수 있습니다. 원격 서버에서 아직 읽지 않은 바이트로 건너뛸 수 없습니다. 이전에 읽은 데이터를 다시 호출할 수 있도록 스트림 데코레이터를

사용하여 PHP 임시 스트림에서 데이터를 버퍼링합니다. 캐시된 데이터의 양이 2MB를 초과할 경우 임시 스트림의 데이터가 메모리에서 디스크로 전송됩니다. `seekable` 스트림 콘텍스트 설정을 사용하여 Amazon S3에서 큰 파일을 다운로드할 때 이 점을 유의하시기 바랍니다.

데이터 업로드

`file_put_contents()`를 사용하여 Amazon S3에 데이터를 업로드할 수 있습니다.

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

`fopen()` 및 "w", "x" 또는 "a" 스트림 액세스 모드로 데이터를 스트리밍하여 큰 파일을 업로드할 수 있습니다. Amazon S3 스트림 래퍼에서는 스트림 읽기와 쓰기(예: "r+", "w+" 등)를 동시에 지원하지 않습니다. HTTP 프로토콜에서 동시 읽기 및 쓰기를 허용하지 않기 때문입니다.

```
$stream = fopen('s3://bucket/key', 'w');
fwrite($stream, 'Hello!');
fclose($stream);
```

> **Note**
>
> Amazon S3는 요청 페이로드를 전송하기 이전에 Content-Length 헤더를 지정해야 합니다. 따라서 스트림이 플러시되거나 닫힐 때까지 PHP 임시 스트림을 사용하여 `PutObject` 작업에서 업로드할 데이터를 내부적으로 버퍼링합니다.

> **Note**
>
> 파일 쓰기 오류는 `fflush`를 호출하는 경우에만 반환되고, 플러시되지 않은 `fclose`를 호출하는 경우에는 반환되지 않습니다. 내부 `fclose`에 대한 응답으로 반환되는 오류에 상관없이 스트림을 닫을 경우 `true`에 대한 반환 값은 `fflush`입니다. PHP에서 구현되는 방법으로 인해 `file_put_contents`를 호출하는 경우에도 이 오류가 반환되지 않습니다.

fopen 모드

PHP의 fopen() 함수를 사용하려면 `$mode` 옵션을 지정해야 합니다. 모드 옵션은 스트림에서 데이터를 읽거나 쓸 수 있는지 여부와 스트림을 열 때 파일이 존재해야 하는지 여부를 지정합니다.

Amazon S3 스트림 래퍼는 Amazon S3 객체를 대상으로 하는 스트림에 대해 다음 모드를 지원합니다.

r	읽기 전용 스트림이며 개체가 이미 존재해야 합니다.
w	쓰기 전용 스트림이며, 개체가 이미 있는 경우 파일을 덮어씁니다.
a	쓰기 전용 스트림이며, 개체가 이미 있는 경우 파일이 임시 스트림에 다운로드되고, 스트림에 쓰는 내용이 이전에 업로드한 데이터에 추가됩니다.
x	쓰기 전용 스트림이며, 개체가 없으면 오류가 발생합니다.

기타 객체 함수

스트림 래퍼를 사용하면 Amazon S3와 같은 사용자 지정 시스템에서 기본 제공된 많은 다른 PHP 함수를 사용할 수 있습니다. 다음은 Amazon S3 스트림 래퍼를 사용하여 Amazon S3에 저장된 객체로 수행할 수 있는 몇 가지 함수입니다.

unlink()	버킷에서 객체를 삭제합니다.
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`
	`DeleteObject` 작업에 사용 가능한 옵션을 전달하여 객체를 삭제하는 방법을 수정할 수 있습니다(예: 특정 객체 버전 지정).
	`// Delete a specific version of an object from a bucket`
	`unlink('s3://bucket/key', stream_context_create([`
	`'s3' => ['VersionId' => '123']`
	`]);`

filesize()	객체의 크기를 가져옵니다.

	`// Get the Content-Length of an object` `$size = filesize('s3://bucket/key', );`

is_file()	URL이 파일인지 확인합니다.
	`if (is_file('s3://bucket/key')) {`
	`  echo 'It is a file!';`
	`}`

file_exists()	객체가 있는지 확인합니다.
	`if (file_exists('s3://bucket/key'))`
	`{`
	`  echo 'It exists!';`
	`}`

filetype()	URL이 파일 또는 버킷(dir)에 매핑되는지 확인합니다.
file()	줄 배열로 객체의 콘텐츠를 로드합니다. `GetObject` 작업에 사용 가능한 옵션을 전달하여 파일을 다운로드하는 방법을 수정할 수 있습니다.
filemtime()	객체를 마지막으로 수정한 날짜를 가져옵니다.
rename()	객체를 복사한 다음 원본을 삭제하여 객체의 이름을 바꿉니다. `CopyObject` 및 `DeleteObject` 작업에 사용 가능한 옵션을 스트림 콘텍스트 파라미터에 전달하여 객체를 복사하고 삭제하는 방법을 수정할 수 있습니다.

> **Note**
>
> copy는 일반적으로 Amazon S3 스트림 래퍼에서 작동하지만, PHP의 내부 copy 함수로 인해 일부 오류가 올바르게 보고되지 않을 수도 있습니다. 대신 AwsS3ObjectCopier의 인스턴스를 사용하는 것이 좋습니다.

버킷 및 폴더 작업

`mkdir()`를 사용하여 버킷 작업

PHP를 사용하여 파일 시스템에서 디렉터리를 생성하고 탐색하는 방법과 비슷하게 Amazon S3 버킷을 생성하고 탐색할 수 있습니다.

다음은 버킷을 생성하는 예제입니다.

```
mkdir('s3://amzn-s3-demo-bucket');
```

> **Note**
>
> 2023년 4월, 이제 Amazon S3가 S3 블록 퍼블릭 액세스 차단을 활성화하고 새로 생성된 모든 버킷에 대해 액세스 제어 목록을 비활성화합니다. 이 변경은 `StreamWrapper`의 `mkdir` 함수가 권한 및 ACL과 함께 작동하는 방식에도 영향을 줍니다. 자세한 내용은 이 AWS 업데이트 내용 문서에서 확인할 수 있습니다.

스트림 콘텍스트 옵션을 `mkdir()` 메서드에 전달하여 CreateBucket 작업에 사용 가능한 파라미터로 버킷을 생성하는 방법을 수정할 수 있습니다.

```
// Create a bucket in the EU (Ireland) Region
mkdir('s3://amzn-s3-demo-bucket', 0500, true,
        stream_context_create([
            's3' => ['LocationConstraint' => 'eu-west-1']
        ]));
```

`rmdir()` 함수를 사용하여 버킷을 삭제할 수 있습니다.

```
// Delete a bucket
rmdir('s3://amzn-s3-demo-bucket);
```

> **Note**
>
> 버킷은 비어있는 경우에만 삭제할 수 있습니다.

`mkdir()`를 사용한 폴더 작업

버킷을 생성하면 파일 시스템에서처럼 폴더 기능을 수행하는 객체를 생성하는 `mkdir()`를 사용할 수 있습니다.

다음 코드 스니펫은 'amzn-s3-demo-bucket'이라는 기존 버킷에 'my-folder'라는 폴더 객체를 추가합니다. 슬래시 (/) 문자를 사용하여 폴더 객체 이름을 버킷 이름 및 추가 폴더 이름과 구분합니다.

```
mkdir('s3://amzn-s3-demo-bucket/my-folder')
```

2023년 4월 이후의 권한 변경에 대한 이전 참고 사항은 폴더 객체를 생성할 때도 적용됩니다. 이 블로그 게시물에는 필요한 경우 권한을 조정하는 방법에 대한 정보가 있습니다.

`rmdir()` 함수를 사용하여 다음 코드 조각에 표시된 것처럼 빈 폴더 객체를 삭제합니다.

```
rmdir('s3://amzn-s3-demo-bucket/my-folder')
```

버킷의 콘텐츠 나열

opendir(), readdir(), rewinddir() 및 closedir() PHP 함수를 Amazon S3 스트림 래퍼와 함께 사용하여 버킷의 콘텐츠를 탐색할 수 있습니다. ListObjects 작업에 사용 가능한 파라미터를 `opendir()` 함수에 사용자 지정 스트림 콘텍스트 옵션으로 전달하여 객체를 나열하는 방법을 수정할 수 있습니다.

```
$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);
}
```

PHP의 RecursiveDirectoryIterator를 사용하여 버킷의 각 객체와 접두사를 재귀적으로 나열할 수 있습니다.

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

일부 HTTP 요청을 포함하는 버킷 콘텐츠를 재귀적으로 나열하는 다른 방법으로 `Aws\recursive_dir_iterator($path, $context = null)` 함수를 사용할 수 있습니다.

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

$iter = Aws\recursive_dir_iterator('s3://bucket/key');
foreach ($iter as $filename) {
    echo $filename . "\n";
}
```

스트림 콘텍스트 옵션

사용자 지정 스트림 콘텍스트 옵션을 전달하여 버킷과 키에 대해 이전에 로드된 정보를 캐시하는 데 사용되는 캐시 또는 스트림 래퍼에 사용되는 클라이언트를 사용자 지정할 수 있습니다.

스트림 래퍼는 모든 작업에서 다음과 같은 스트림 콘텍스트 옵션을 지원합니다.

`client`

명령을 실행하는 데 사용할 `Aws\AwsClientInterface` 객체입니다.

`cache`

이전에 가져온 파일 통계를 캐시하는 데 사용할 `Aws\CacheInterface`의 인스턴스입니다. 기본적으로 스트림 래퍼는 인 메모리 LRU 캐시를 사용합니다.

파일 및 디렉터리 전송

AWS SDK for PHP 버전 3은 Amazon S3와 파일 및 디렉터리를 주고받는 두 가지 접근 방식을 제공합니다. 두 솔루션 모두 대용량 파일에 대한 멀티파트 업로드 및 다운로드의 복잡성을 처리하지만 설계 철학, 기능 세트 및 사용 패턴은 다릅니다.

전송 옵션 개요

애플리케이션의 요구 사항에 가장 적합한 전송 접근 방식을 선택합니다.

[S3 Transfer Manager](#)(권장)

파일 전송을 위한 포괄적인 솔루션을 제공하는 최신 상위 수준 라이브러리입니다. 광범위한 구성 옵션, 기본 제공 진행 상황 추적, 사용자 지정 다운로드 핸들러 및 강력한 오류 처리를 제공합니다. S3 Transfer Manager는 promise 기반 API를 사용하며 고급 필터링 기능을 통해 개별 파일 작업과 디렉터리 전송을 모두 지원합니다.

[전송](#)

특히 대량 디렉터리 작업에 초점을 맞춘 디렉터리 전송 구현입니다. 기본 구성 옵션으로 전체 디렉터리를 업로드하고 다운로드할 수 있는 더 간단한 API를 제공합니다. 이 접근 방식은 S3 Transfer Manager에 비해 기능이 적습니다.

주요 차이점

다음 표에서는 두 전송 접근 방식의 주요 차이점을 강조합니다.

기능	S3 전송 관리자	Transfer
개별 파일 작업	예(단일 파일 업로드/다운로드)	아니요(디렉터리 작업만 해당)
디렉터리 작업	예(고급 필터링 사용)	예(기본 디렉터리 전송)
진행 상황 추적	사용자 지정 리스너가 기본 제공	제한적(디버그 출력만 해당)
사용자 지정 다운로드 핸들러	예	아니요
체크섬 검증	구성을 사용한 자동	수동(add_content_md5 옵션)
오류 처리	장애 정책과 함께 포괄적	기본 promise 기반 처리
구성 옵션	광범위한(8개 이상의 옵션)	기본(6개 옵션)
API 설계	요청/응답 객체	간단한 생성자 파라미터

올바른 접근 방식 선택

필요한 경우 S3 Transfer Manager를 사용합니다.

- 개별 파일 업로드 또는 다운로드 작업
- 고급 진행 상황 추적 및 모니터링
- 특수 처리를 위한 사용자 지정 다운로드 핸들러
- 포괄적인 오류 처리 및 재시도 정책
- 멀티파트 작업에 대한 세분화된 제어
- 복잡한 필터링 로직을 사용한 디렉터리 작업

필요한 경우 전송을 사용합니다.

- S3 간 간단한 디렉터리 전송
- 최소 구성 및 설정
- Transfer를 사용하는 기존 코드와의 호환성
- 기본 멀티파트 업로드 기능

> **Note**
>
> 새 애플리케이션의 경우 파일 전송을 위한 보다 포괄적이고 유연한 솔루션을 제공하므로 S3 Transfer Manager를 사용하는 것이 좋습니다.

S3 전송 관리자

S3 Transfer Manager는 Amazon S3에서 파일을 업로드하고 다운로드할 수 있는 인터페이스를 제공합니다. 단일 파일 작업 또는 디렉터리 작업에 사용할 수 있습니다.

Transfer Manager는 멀티파트 업로드 및 다운로드를 자동으로 처리합니다. 대용량 파일을 관리하고 진행 상황을 추적합니다. 라이브러리는 S3 데이터 전송에 대한 모범 사례를 구현합니다. 이를 사용하여 PHP 애플리케이션에서 파일 전송 기능을 구축할 수 있습니다.

주요 기능

S3 Transfer Manager는 다음과 같은 주요 기능을 제공합니다.

- 간단한 API: 파일 및 디렉터리 업로드 및 다운로드
- 사용자 지정 다운로드 핸들러: 자체 다운로드 로직 구현
- 자동 멀티파트 업로드 및 다운로드: 대용량 파일을 자동으로 처리
- 동시 처리: 처리량 극대화
- 진행 상황 추적: 전송 상태 모니터링
- 사용자 지정 가능한 동작: 광범위한 옵션 구성
- 오류 처리: 재시도 및 실패 정책 구성
- 디렉터리 작업: 여러 파일을 배치로 전송
- 체크섬 검증: 데이터 무결성 보장

설치

S3 Transfer Manager는 AWS SDK for PHP 버전 3에 포함되어 있습니다. 별도로 설치할 필요가 없습니다.

Composer를 사용하여를 설치하려면 다음 명령을 실행합니다.

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

이 명령은 S3 Transfer Manager를 포함하여 전체 AWS SDK for PHP 버전 3을 설치합니다.

기본 사용법

다음 예제에서는 S3 Transfer Manager를 사용하는 방법을 보여줍니다.

```
<?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

S3 Transfer Manager를 사용하여 기본 Amazon S3 클라이언트를 생성하는 경우 고객은 Transfer Manager `config` 옵션의 `default_region` 파라미터를 사용하여 클라이언트의 기본 리전을 전달할 수 있습니다. 그렇지 않으면 Amazon S3 클라이언트는 구성 해결을 위한 기본 동작을 사용하여 리전을 확인하려고 시도하고 리전이 확인되지 않으면 예외가 발생합니다.

전송 관리자 생성

다음 두 가지 방법으로 전송 관리자를 생성할 수 있습니다.

기존 S3 클라이언트 사용

기존 `S3Client` 인스턴스를 `S3TransferManager` *<## ##>* 생성자에 전달합니다.

```
<?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);
```

기본 S3 클라이언트 생성 사용

클라이언트null로 전달하고 구성 옵션을 지정합니다.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);
```

구성

S3 Transfer Manager는 구성 옵션을 수락하여 동작을 사용자 지정합니다. 전송 관리자의 인스턴스를 생성할 때 이러한 옵션을 제공합니다. 구성 파라미터는 S3TransferManagerConfig <## ##>의 배열 또는 인스턴스일 수 있습니다.

다음 예제에서는 S3 Transfer Manager 인스턴스를 구성합니다.

```
<?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**
>
> 구성을에 배열로 제공하면 SDKS3TransferManager는 내부적으로를 호출S3TransferManagerConfig::fromArray하여 적절한 유형으로 변환합니다.

구성 옵션

모든 구성 옵션은 선택 사항이며 지정하지 않으면 기본값을 사용합니다.

옵션	Type	기본값	설명
`target_part_size_bytes`	int	8MB	멀티파트 업로드/다운로드의 최소 파트 크기입니다.
`multipart_upload_threshold_bytes`	int	16MB	멀티파트 업로드를 사용하기 위한 파일 크기 임계값입니다.
`request_checksum_calculation`	문자열	'when_supported'	체크섬 계산을 활성화합니다. 유효한 값은 'when_supported', 'when_required'입니다.
`response_checksum_validation`	문자열	'when_supported'	객체를 가져올 때 체크섬 검증을 활성화합니다. 유효한 값은 'when_supported', 'when_required'입니다.
`multipart_download_type`	문자열	'파트'	대용량 파일에 대한 다운로드 전략입니다. 유효한 값은 'part'(멀티파트 다운로드), 'range'(범위 요청)입니다.
`concurrency`	int	5	최대 동시 작업 수입니다.
`track_progress`	bool	FALSE	전송 진행 상황을 추적할지 여부입니다.
`default_region`	문자열	'us-east-1'	AWS 리전 S3 클라이언트가 제공되지 않은 경우를 사용합니다.

파일 작업

S3 Transfer Manager는 개별 파일을 업로드하고 다운로드하는 방법을 제공합니다.

로컬 파일 업로드

Amazon S3에 파일을 업로드하려면 `upload <## ##>` 메서드를 사용합니다.

```php
<?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();
```

upload 메서드 파라미터

upload 메서드는 UploadRequest <## ##>의 인스턴스를 인수로 수락합니다.

UploadRequest 파라미터

파라미터	Type	필수	설명
$source	문자열|StreamInterface	예	업로드할 데이터가 포함된 로컬 파일 경로 또는 스트림의 경로입니다.
$uploadRequestArgs	array	예	요청 인수를 업로드합니다(버킷 및 키를 포함해야 함).
$config	array	아니요	이 업로드와 관련된 구성 재정의. 구성 옵션에 대한 자세한 내용은 다음 섹션을 참조하세요.
$listeners	array	아니요	이 업로드를 모니터링하기 위한 TransferListener 객체 배열입니다.
$progressTracker	TransferListener	아니요	이 업로드에 대한 진행률 트래커입니다.

$config에 대한 옵션 **UploadRequest**

SDK는 S3 Transfer Manager의 구성에서 기본$config값을 확인합니다.

옵션	Type	필수	설명
multipart_upload_t	int	아니요	멀티파트 업로드가 트리거될 때의 기본 임계값을 재정의합니다.

옵션	Type	필수	설명
hreshold_bytes			
target_part_size_bytes	int	아니요	기본 대상 파트 크기를 바이트 단위로 재정의합니다.
track_progress	bool	아니요	진행 상황 추적을 활성화하는 기본 옵션을 재정의합니다. 이 옵션이 true 이고 progressTracker 파라미터를 제공하지 않으면 SDK는 기본 구현을 사용합니다.
concurrency	int	아니요	동시성의 기본값을 재정의합니다.
request_checksum_calculation	문자열	No	요청 체크섬 계산을 수행해야 하는지 여부에 대한 기본값을 재정의합니다.

upload 메서드가 성공적으로 실행되면 UploadResult <## ##>를 반환합니다.

S3 객체 다운로드

Amazon S3에서 객체를 다운로드하려면 download <## ##> 메서드를 사용합니다.

```
<?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();
```

download 메서드 파라미터

download 메서드는 DownloadRequest <## ##>의 인스턴스를 인수로 수락합니다.

DownloadRequest 파라미터

파라미터	Type	필수	설명
$source	string|array|null	아니요	S3에서 다운로드할 객체입니다. 이 파라미터를 S3 URI 문자열("s3://amzn-s3-demo-bucket/key"), 버킷 및 키 키가 있는 배열 또는 로 제공할 수 있습니다null. 이 값이 이면 $download

파라미터	Type	필수	설명
			`RequestArgs` 파라미터의 버킷 및 키 값을 `null`전달합니다.
`$downloadRequestArgs`	array	아니요	객체 요청 인수를 추가로 다운로드합니다. 이 `$source`인 경우 여기에 버킷 및 키 값을 `null`입력합니다.
`$config`	array	아니요	이 다운로드와 관련된 구성 재정의. 구성 옵션에 대한 자세한 내용은 다음 섹션을 참조하세요.
`$downloadHandler`	`AbstractDownloadHandler <## ##>|null`	아니요	각 객체 청크를 수신하고 다운로드를 관리하는 핸들러입니다. `download` 메서드의 기본 핸들러는 스트림 기반 핸들러를 사용합니다. 이 핸들러는 각 객체 청크를 스트림으로 푸시합니다. `downloadFile` 메서드의 기본 핸들러(참조the section called “로컬 파일에 S3 객체 다운로드”)는 파일 기반 핸들러를 사용합니다. 이 핸들러는 각 청크를 파일에 씁니다. 자체를 구현`DownloadHandler` 하여 SDK가 다운로드한 데이터를 저장하는 위치와 방법을 사용자 지정할 수 있습니다. 예를 들어 파일이나 스트림 대신 데이터베이스, 클라우드 스토리지 또는 사용자 지정 처리 파이프라인에 데이터를 저장할 수 있습니다.
`$progressTracker`	`TransferListener`	아니요	이 다운로드에 대한 진행률 트래커입니다.

파라미터	Type	필수	설명
`$listeners`	`array`	아니요	이 다운로드를 모니터링하기 위한 `AbstractTransferListener` 객체 배열입니다.

`$config`에 대한 옵션 `DownloadRequest`

SDK는 S3 Transfer Manager의 구성에서 기본`$config`값을 확인합니다.

옵션	Type	필수	설명
`multipart_download_type`	문자열	No	기본 멀티파트 다운로드 유형을 재정의합니다. 유효한 값은 'part', 'range'입니다.
`response_checksum_validation`	문자열	No	체크섬 검증을 수행해야 하는지 여부에 대해 전송 관리자 구성에서 확인된 값을 재정의합니다. SDK는 `$getObjectRequestArgs` 의에 이 `ChecksumMode` 없는 경우에만 이 옵션을 고려합니다`DownloadRequest` .
`track_progress`	bool	아니요	진행 상황 추적을 활성화하기 위한 기본 옵션을 재정의합니다. 이 옵션이 `true` 이고 SDK에서 기본 구현`DownloadRequest`, 을 사용하는 `progressTracker` 파라미터를 제공하지 않는 경우. `$progressTracker` 파라미터가 일 때 기본 진행률 트래커를 활성화하려면이 옵션을 사용합니다`null`.

옵션	Type	필수	설명
`target_part_size_bytes`	int	아니요	범위 멀티파트 다운로드에 사용할 바이트 단위의 파트 크기입니다. 이 파라미터를 제공하지 않으면 다운로드는 전송 관리자에 `target_part_size_bytes` 구성된를 사용합니다.

download 메서드가 성공적으로 실행되면 DownloadResult <## ##>를 반환합니다.

로컬 파일에 S3 객체 다운로드

S3 객체를 로컬 파일로 직접 다운로드하려면 downloadFile 메서드를 사용합니다.

```
<?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";
```

downloadFile 메서드 파라미터

downloadFile 메서드는의 인스턴스를 인수DownloadFileRequest로 수락합니다.

DownloadFileRequest 파라미터

옵션	Type	필수	설명
$destination	문자열	예	파일이 저장되는 로컬 경로입니다.
$failsWhenDestinationExists	bool	예	대상 파일이 이미 있는 경우 실패할지 여부입니다. 이 옵션이 false 이고 대상 파일이 있는 경우 SDK는 기존 파일을 삭제하거나 재정의합니다.
$downloadRequest	DownloadRequest	예	구성된 DownloadRequest 객체입니다. 자세한 내용은 DownloadRequest 객체를 참조하세요.

downloadFile 메서드가 성공적으로 실행되면 DownloadResult <## ##>를 반환합니다.

디렉터리 작업

S3 Transfer Manager는 전체 디렉터리 전송을 처리할 수 있습니다.

디렉터리 업로드

S3 Transfer Manager는 전체 디렉터리를 S3 버킷에 업로드할 수 있습니다. < #### # # ## ## ## ###?>

```
<?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";
```

uploadDirectory 메서드 파라미터

uploadDirectory 메서드는 UploadDirectoryRequest<## ##>의 인스턴스를 인수로 허용합니다.

UploadDirectoryRequest 파라미터

파라미터	Type	필수	설명
$sourceDirectory	문자열	예	업로드할 로컬 디렉터리의 경로입니다.
$targetBucket	문자열	예	대상 S3 버킷 이름입니다.
$uploadRequestArgs	array	아니요	모든 파일에 대한 추가 업로드 객체 요청 파라미터입니다.
$config	array	아니요	디렉터리 업로드를 위한 구성 옵션입니다. 구성 옵션에 대한 자세한 내용은 다음 섹션을 참조하세요.
$listeners	array	아니요	TransferListener 객체의 배열입니다. SDK는 파일TransferListener 당 각를 복제합니다.
$max_depth	bool	-1 "런타임 기본값"	재귀 파일의 최대 깊이를 나타냅니다.
$progressTracker	TransferListener	아니요	모든 업로드에 대한 진행률 트래커입니다.

$config에 대한 옵션 **UploadDirectoryRequest**

옵션	Type	기본값	설명
recursive	bool	false	하위 디렉터리를 재귀적으로 업로드할지 여부입니다.
follow_symbolic_links	bool	false	심볼 링크를 따를지 여부입니다.
s3_prefix	문자열	''	모든 객체 키에 추가할 접두사입니다.
s3_delimiter	문자열	'/'	S3 객체 키에 사용할 구분 기호입니다.
track_progress	bool	false	진행 상황을 추적할지 여부입니다.
max_concurrency	int	100	최대 동시 업로드 수입니다.
filter	호출 가능	null	업로드할 파일을 필터링하는 함수입니다.
upload_object_request_modifier	호출 가능	null	각 업로드 요청을 사용자 지정하는 함수입니다.
failure_policy	호출 가능	null	업로드 실패를 처리하는 함수입니다.

호출 가능한 **$config** 옵션 유형

옵션 이름	파라미터 이름	파라미터 유형	파라미터 정보
filter	$file	SplFileInfo |문자열	문자열로 캐스팅되는 경우 파일 경로입니다. 그렇지 않으면의 인스턴스입니다SplFileInfo .
upload_object_request_modifier	$requestArgs	array	각 개별 업로드 요청에 대한 요청 인수입니다. 배열 옵션에 대한 자세한 내용은 putObject 메서드의 파라미터 구문 섹션을 참조하세요.
failure_policy	$uploadRequestArgs	array	각 개별 업로드 요청에 대한 요청 인수입니다. 배열 옵션에 대한 자세한 내용은 배열 옵션에 대한 putObject 메서드의 파라미터 구문 섹션을 참조하세요.
	$uploadDirectoryArgs	array	업로드 디렉터리 작업을 위한 소스 디렉터리와 대상 버킷이 있는 배열입니다.
	$reason	Throwable |문자열	실패 원인에 대한 정보가 포함된 예외 소유자입니다.
	$uploadDirectoryResult	UploadDirectoryResult	성공적으로 업로드된 객체 수와 실패한 객체 수가 포함된 객체입니다.

uploadDirectory 메서드가 성공적으로 실행되면 UploadDirectoryResult *<## ##>*를 반환합니다.

디렉터리 다운로드

S3 버킷에서 디렉터리를 다운로드할 수 있습니다. *<### ##### # # ## ### #####? ## #### # # ## ### #### ###?>*

```
<?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";
```

downloadDirectory 메서드 파라미터

downloadDirectory 메서드는 DownloadDirectoryRequest<## ##> 인스턴스를 인수로 허용합니다.

DownloadDirectoryRequest 파라미터

파라미터	Type	필수	설명
$sourceBucket	문자열	예	객체를 다운로드할 소스 S3 버킷 이름입니다.
$destinationDirectory	문자열	예	파일을 다운로드할 로컬 디렉터리입니다.
$downloadRequestArgs	array	아니요	모든 파일에 대한 추가 객체 다운로드 요청 파라미터입니다.
$config	array	아니요	디렉터리 다운로드를 위한 구성 옵션입니다. 구성 옵션에 대한 자세한 내용은 다음 섹션을 참조하세요.
$listeners	array	아니요	TransferListener 객체의 배열입니다. SDK는 파일TransferListener 당 각를 복제합니다.

파라미터	Type	필수	설명
`$progressTracker`	`TransferListener`	아니요	모든 다운로드에 대한 진행률 트래커입니다.

`$config`에 대한 옵션 `DownloadDirectoryRequest`

옵션	Type	기본값	설명
`s3_prefix`	문자열	"	필터링을 위한 접두사입니다(에 없는 경우`list_objects_v2_args`).
`track_progress`	bool	`false`	진행 상황을 추적할지 여부입니다.
`filter`	호출 가능	`null`	다운로드할 S3 객체를 필터링하는 함수입니다.
`download_object_request_modifier`	호출 가능	`null`	각 다운로드 요청을 사용자 지정하는 함수입니다.
`list_objects_v2_args`	array	[]	`ListObjectsV2` 작업에 대한 인수입니다. 이 작업은 다운로드할 객체의 메타데이터를 가져옵니다.
`fails_when_destination_exists`	bool	false	객체 대상 파일 경로가 이미 있는 경우 실패할지 여부입니다.
`failure_policy`	호출 가능	null	다운로드 실패를 처리하는 함수입니다.
`max_concurrency`	int	100	최대 동시 다운로드 수입니다.

옵션	Type	기본값	설명
`target_part_size_bytes`	int	varies	멀티파트 다운로드 유형의 범위가 지정될 때 멀티파트 다운로드의 파트 크기입니다.

호출 가능한 **`$config`** 옵션 유형

옵션 이름	파라미터 이름	파라미터 유형	파라미터 정보
`filter`	`$objectKey`	문자열	버킷의 객체 키 이름입니다.
`download_object_request_modifier`	`$requestArgs`	array	각 개별 다운로드 요청에 대한 요청 인수입니다. 배열 옵션에 대한 자세한 내용은 getObject 메서드의 파라미터 구문 섹션을 참조하세요.
`failure_policy`	`$downloadObjectRequestArgs`	array	각 개별 다운로드 요청에 대한 요청 인수입니다. 배열 옵션에 대한 자세한 내용은 getObject 메서드의 파라미터 구문 섹션을 참조하세요.
	`$downloadDirectoryRequest`	array	디렉터리 다운로드 작업을 위한 소스 버킷과 대상 디렉터리가 있는 배열입니다.
	`$reason`	`Throwable`	실패 원인에 대한 정보가 포함된 예외 소유자입니다.

옵션 이름	파라미터 이름	파라미터 유형	파라미터 정보
	`$downloadDirectoryResult`	DownloadDirectoryResult <## ##>	성공적으로 다운로드한 객체 수와 실패한 객체 수가 포함된 객체입니다.

디렉터리 작업 작동 방식

이 섹션에서는 S3 Transfer Manager가 디렉터리 작업 중에 파일 경로와 객체 키를 처리하는 방법을 설명합니다.

업로드 경로 처리

디렉터리를 업로드하면 SDK는 파일 경로에서 S3 객체 키를 구성합니다.

1. 소스 디렉터리를 기준으로 각 파일의 경로를 계산합니다.
2. 지정된 경우 `s3_prefix` 값을 준비합니다(후행 슬래시 자동 추가).
3. OS 디렉터리 구분자를 `s3_delimiter` (기본값 /)로 바꿉니다.

`recursive` 옵션은 하위 디렉터리가 포함되는지 여부를 제어합니다. `false` (기본값)인 경우 최상위 파일만 업로드됩니다. 이면 하위 디렉터리의 `true`모든 파일이 업로드되어 객체 키의 디렉터리 구조가 보존됩니다.

> **Note**
>
> 업로드된 각 파일에 대한 S3의 키는 제공된 s3 접두사이며, 기본적으로 null과 지정된 소스 디렉터리에서 시작하는 파일의 상대 경로입니다. 또한 확인된 키 이름의 디렉터리 구분자를 제공된 s3 구분 기호로 바꿉니다. 기본적으로 /입니다.
> 예제 1.
>
> ```
> $sourceDirectory = "/opt/my-upload-directory";
> $s3Prefix = "important/";
> $s3Delimiter = "/"; // Default value
> ```
>
> 디렉터리 구조

```
/opt/my-upload-directory/
-- my-file1.txt
-- my-file-2.txt
-- sub-dir/
---- my-another-file1.txt
---- my-another-file2.txt
```

각 파일의 키는 다음과 같습니다.

```
$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
```

예제 2.

```
$sourceDirectory = "/opt/my-docs";
$s3Prefix = ''; // No provided and it will defaulted to empty string
$s3Delimiter = "/"; // Default value
```

디렉터리 구조

```
/opt/my-docs/
-- my-file1.txt
-- my-file-2.txt
-- sub-dir/
---- my-another-file1.txt
---- my-another-file2.txt
```

각 파일의 키는 다음과 같습니다.

```
$s3Prefix + $fileRelativePath;
```

```
- my-file1.txt
- my-file-2.txt
```

```
- sub-dir/my-another-file1.txt
- sub-dir/my-another-file2.txt
```

Example디렉터리 경로 처리 업로드

```
<?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
```

다운로드 경로 처리

디렉터리를 다운로드할 때 SDK는 S3 객체 키에서 로컬 파일 경로를 구성합니다.

1. 지정된 경우 객체 키`s3_prefix`에서를 제거합니다.
2. S3 구분 기호(/)를 OS 디렉터리 구분자로 바꿉니다.
3. 대상 디렉터리에 결과를 추가합니다.

다운로드는 항상 재귀적입니다. SDK는 지정된 접두사 아래의 모든 객체를 검색하고 필요에 따라 하위 디렉터리를 생성합니다. 보안을 위해 대상 디렉터리 외부에서 경로가 확인되지 않는지 확인합니다.

Example디렉터리 경로 처리 다운로드

```
<?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

다운로드한 객체의 파일 경로는 다음과 같이 확인됩니다.

- 제공된 s3 접두사는 객체 키에서 제거됩니다.
- s3Delimiter가 OS 디렉터리 구분 기호와 다른 경우 s3 구분 기호는 OS 디렉터리 구분 기호로 대체됩니다.

예제 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/";
```

이러한 객체가 다운로드되면 다음과 같이 배치됩니다.

```
/opt/bucket/downloads/
-- file-1.txt
-- file-2.txt
-- subdir/
---- file-1.txt
```

보시다시피 접두사는 객체의 최종 파일 경로에서 제거되었습니다.
예제 2 - s3 접두사 없음

```
$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/";
```

이러한 객체가 다운로드되면 다음과 같이 배치됩니다.

```
/opt/bucket/downloads/
-- README.md
-- my-docs/
---- file-1.txt
---- file-2.txt
---- statements/
------ file-1.txt
```

전송 리스너

다양한 전송 이벤트에 반응하는 자체 리스너를 구현할 수 있습니다. 이벤트는 다음과 같습니다.

- **transferInitiated**: 전송이 시작될 때
- **bytesTransferred**: 전송이 소스에서 대상으로 바이트를 전송하는 경우
- **transferComplete**: 오류 없이 전송이 완료된 경우
- **transferFail**: 완료 전에 전송이 실패한 경우

각 S3TransferManager 작업(예: uploadFile 및 downloadDirectory)에는 listeners 파라미터가 있습니다. 이 파라미터는 각 항목을의 인스턴스로 예상하는 배열입니다AbstractTransferListener.

AbstractTransferListener는 전송 작업을 모니터링하기 위해 확장할 수 있는 추상적 기본 클래스입니다. 모든 전송 이벤트에 대해 빈 메서드 구현을 제공합니다. 처리하려는 이벤트의 메서드만 재정의하면 됩니다.

각 AbstractTransferListener 메서드는 $context 인수를 수신합니다. 이 인수에는 작업에 대한 키-값 정보가 포함되어 있습니다. 컨텍스트에는 다음이 포함됩니다.

- request_args (배열) - 요청 파라미터
- progress_snapshot (TransferProgressSnapshot<## ##> 인스턴스) - 현재 요청 상태
- reason (의 인스턴스Throwable) - transferFail 메서드에 대해서만 제공됨

progress_snapshot 속성에는 실패 이벤트에 대한 reason 속성도 포함됩니다. $context['reason'] 또는를 통해 예외에 액세스할 수 있습니다$context['progress_snapshot']#getReason().

다음 코드는 AbstractTransferListener 클래스 구조를 보여줍니다.

```
<?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 {}
}
```

다음 예제에서는 `transferInitiated` 및 `transferFail` 메서드만 재정의하는 `AbstractTransferListener` 구현을 보여줍니다.

```
<?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();
```

진행 상황 추적

S3 Transfer Manager는 내장된 진행 상황 추적 기능을 제공합니다.

모든 작업 추적

`S3TransferManager` 인스턴스`true`에서를 로 설정하여 모든 작업(파일 및 디렉터리)에 대한 추적을 활성화`track_progress`할 수 있습니다.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

```
$transferManager = new S3TransferManager(
    null,
    [
        'track_progress' => true,
    ]
);
```

이 설정을 사용하면 SDK는 파일 작업에 내장 SingleProgressTracker<## ##>를 사용하고 디렉터리 작업에 MultiProgressTracker<링크 추가>를 사용합니다.

단일 파일 작업 추적

전송 관리자에서 진행 상황 추적을 활성화하지 않은 경우 두 가지 방법으로 개별 메서드에 대한 추적을 활성화할 수 있습니다.

특정 파일 작업의 $config 배열true에서 track_progress를 로 설정합니다.

```
<?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,
        ],
        []
    )
);
```

새 SingleProgressTracker 인스턴스를 생성하거나의 사용자 지정 구현을 제공합니다AbstractTransferListener. 트래커를 $progressTracker 파라미터로 파일 작업에 전달합니다.

```
<?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.
    )
);
```

다음 예제에서는 라는 파일의 업로드 진행 상황에 SingleProgressTracker 대한 기본 제공의 샘플 콘솔 출력을 보여줍니다MyFile.txt.

```
MyFile.txt:
[##########################                                  ] 50% 5242880/10485760 B
```

단일 디렉터리 작업 추적

전송 관리자에서 진행 상황 추적을 활성화하지 않은 경우 두 가지 방법으로 디렉터리 작업에 대해 활성화할 수 있습니다.

메서드의 $config 배열 인수true에서 'track_progress'를 로 설정합니다.

```
<?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,
        ],
        []
    )
);
```

새 MultiProgressTracker 인스턴스를 생성하고 디렉터리 작업에 $progressTracker 파라미터로 전달합니다.

```
<?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()
    )
);
```

다음 예제는 디렉터리의 업로드 진행 상황에 MultiProgressTracker 대한 기본 제공의 샘플 콘솔 출력을 보여줍니다.

```
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
```

각 파일의 진행 상황은 마지막 줄의 요약 진행 상황과 함께 나열됩니다.

사용자 지정 진행 상황 추적

파일 작업에 대한 자체 진행 상황 추적을 구현할 수 있습니다. AbstractTransferListener 인터페이스를 확장하는 클래스를 생성하고 요청 시 메서드에 전달합니다.

다음 구현에서는 단일 파일 작업에 대한 진행률 정보를 텍스트 형식으로 표시합니다.

```
<?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()
    )
);
```

다음 예제에서는 진행 중인 파일 업로드를 위해 콘솔에 대한 샘플 출력을 보여줍니다.

```
Starting transfer of file.txt...
Progress: 42% (16.8MB of 40MB)
```

기본 제공 진행률 트래커 사용자 지정

다음과 같은 몇 가지 방법으로 진행 상황 추적을 사용자 지정할 수 있습니다.

- `ProgressTrackerInterface<## ##>`를 확장`AbstractTransferListener`하고 구현하여 사용자 지정 구현을 생성합니다.
- 내장 트래커에 전달되는 `ConsoleProgressBar<## ##>`를 구성하여 기존 구현을 사용자 지정합니다.
- 사용자 지정 `ProgressBar<## ##>`를 구현하여 내장 트래커에 전달합니다.

다음 예제에서는 진행 상황 추적을 사용자 지정하는 방법을 보여줍니다. SDK의 진행 상황 추적 구성 요소 간의 관계를 보려면 클래스 다이어그램을 참조하세요.

사용자 지정 **`ConsoleProgressBar`**

기본 막대 형식 사용자 지정

다음 예제에서는 기본 진행률 표시줄 형식인 `ColoredTransferProgressBarFormat<## ##>`를 사용자 지정합니다. 이 형식은 전송 상태에 따라 다른 색상의 진행 상황을 보여줍니다.

이 예제는 다음과 같이 변경됩니다.

- 기본 "#"에서 "="까지의 문자
- 기본값 50~100의 막대 너비

```
<?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);
```

다음 이미지는 너비ConsoleProgressBar가 100이고 문자가 "="인 사용자 지정 내장에 대한 콘솔 출력을 보여줍니다.

```
key:
[                                                  ] 0% 0/1048576 B

key:
[==================================================] 100% 1048576/1048576 B
```

일반 진행률 표시줄 형식을 사용하여 사용자 지정

다음 예제에서는 기본 제공 PlainProgressBarFormat<## ##>를 사용자 지정 문자 및 너비와 함께 사용하여 기본 진행률 표시줄을 사용자 지정합니다. 는 검은색으로만 PlainProgressBarFormat 표시됩니다. 전송 상태에 따라 색상이 변경되지 않습니다.

이 예제는 다음과 같이 변경됩니다.

- 기본 "#"에서 "*"까지의 문자
- 기본값 50~25의 막대 너비
- 에서 ColoredTransferProgressBarFormat로 기본 progressBarFormat 인스턴스 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\S3Transfer\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);
```

다음 이미지는 너비가 25이고 문자가 "*"인 일반 진행률 표시줄 형식을 ConsoleProgressBar 사용하는 사용자 지정 내장에 대한 콘솔 출력을 보여줍니다.

새 막대 형식을 생성하여 사용자 지정

다음 예제에서는 기본 제공 형식을 사용자 지정하는 대신 새 막대 형식(ProgressBarFormat<## ## > 확장)을 사용하여 기본 진행률 표시줄을 사용자 지정합니다.

```
<?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);
```

다음 이미지는 사용자 지정 진행률 표시줄 형식을 ConsoleProgressBar 사용하는 사용자 지정 내장에 대한 콘솔 출력을 보여줍니다.

```
key - 0% [0 Bytes]
key - 11% [1153434 Bytes]
key - 55% [5767168 Bytes]
key - 100% [10485760 Bytes]
```

사용자 지정 막대 형식은 여러 파라미터를 허용합니다.

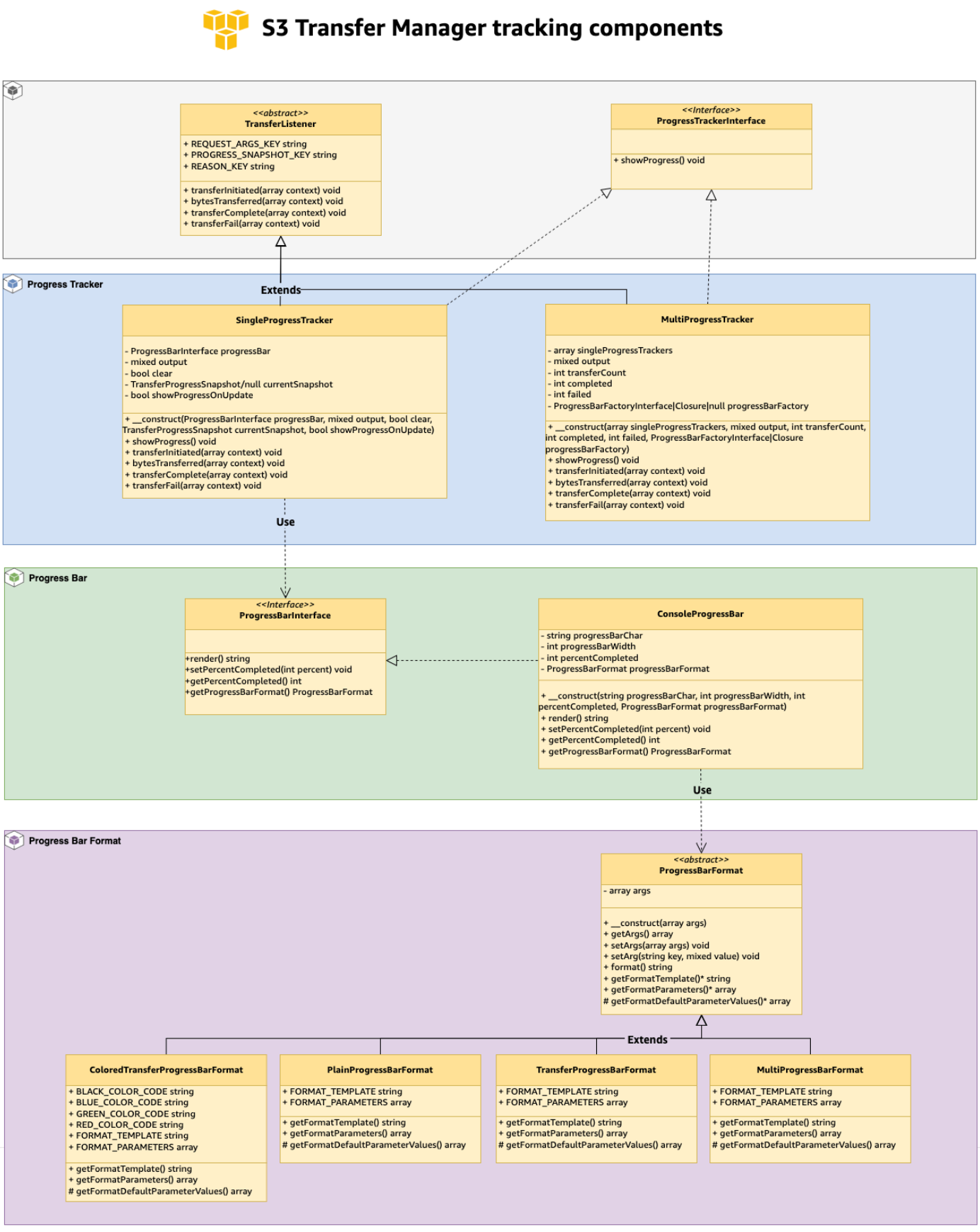
- 객체 이름
- %
- progress_bar
- to_be_transferred
- 전송됨
- 단위

구분된 (파이프 문자) 파라미터 이름을 결합하여 출력을 사용자 지정할 수 있습니다. 이 예제에서는 다음 템플릿을 사용합니다. PercentBarFormat 클래스의 getFormatTemplate 메서드는 다음 템플릿을 반환합니다.

```
"|object_name| - |percent|% [|transferred| |unit|ytes]"
```

클래스 다이어그램: S3 Transfer Manager 추적 구성 요소

다음 클래스 다이어그램은 S3 Transfer Manager에서 추적 구성 요소가 함께 작동하는 방식을 보여줍니다. 이러한 관계를 이해하면 사용자 지정 리스너 및 진행 상황 추적을 구현하는 데 도움이 됩니다.


S3 Transfer Manager tracking components
<<abstract>>
TransferListener
+ REQUEST_ARGS_KEY string
+ PROGRESS_SNAPSHOT_KEY string
+ REASON_KEY string
+ transferInitiated(array context) void
+ bytesTransferred(array context) void
+ transferComplete(array context) void
+ transferFail(array context) void
<<Interface>>
ProgressTrackerInterface
+ showProgress() void
Progress Tracker
Extends
SingleProgressTracker
- ProgressBarInterface progressBar
- mixed output
- bool clear
- TransferProgressSnapshot/null currentSnapshot
- bool showProgressOnUpdate
+ __construct(ProgressBarInterface progressBar, mixed output, bool clear, TransferProgressSnapshot currentSnapshot, bool showProgressOnUpdate)
+ showProgress() void
+ transferInitiated(array context) void
+ bytesTransferred(array context) void
+ transferComplete(array context) void
+ transferFail(array context) void
MultiProgressTracker
- array singleProgressTrackers
- mixed output
- int transferCount
- int completed
- int failed
- ProgressBarFactoryInterface|Closure|null progressBarFactory
+ __construct(array singleProgressTrackers, mixed output, int transferCount, int completed, int failed, ProgressBarFactoryInterface|Closure progressBarFactory)
+ showProgress() void
+ transferInitiated(array context) void
+ bytesTransferred(array context) void
+ transferComplete(array context) void
+ transferFail(array context) void
Use
Progress Bar
<<Interface>>
ProgressBarInterface
+render() string
+setPercentCompleted(int percent) void
+getPercentCompleted() int
+getProgressBarFormat() ProgressBarFormat
ConsoleProgressBar
- string progressBarChar
- int progressBarWidth
- int percentCompleted
- ProgressBarFormat progressBarFormat
+ __construct(string progressBarChar, int progressBarWidth, int percentCompleted, ProgressBarFormat progressBarFormat)
+ render() string
+ setPercentCompleted(int percent) void
+ getPercentCompleted() int
+ getProgressBarFormat() ProgressBarFormat
Use
Progress Bar Format
<<abstract>>
ProgressBarFormat
- array args
+ __construct(array args)
+ getArgs() array
+ setArgs(array args) void
+ setArg(string key, mixed value) void
+ format() string
+ getFormatTemplate()* string
+ getFormatParameters()* array
# getFormatDefaultParameterValues()* array
Extends
ColoredTransferProgressBarFormat
+ BLACK_COLOR_CODE string
+ BLUE_COLOR_CODE string
+ GREEN_COLOR_CODE string
+ RED_COLOR_CODE string
+ FORMAT_TEMPLATE string
+ FORMAT_PARAMETERS array
+ getFormatTemplate() string
+ getFormatParameters() array
# getFormatDefaultParameterValues() array
PlainProgressBarFormat
+ FORMAT_TEMPLATE string
+ FORMAT_PARAMETERS array
+ getFormatTemplate() string
+ getFormatParameters() array
# getFormatDefaultParameterValues() array
TransferProgressBarFormat
+ FORMAT_TEMPLATE string
+ FORMAT_PARAMETERS array
+ getFormatTemplate() string
+ getFormatParameters() array
# getFormatDefaultParameterValues() array
MultiProgressBarFormat
+ FORMAT_TEMPLATE string
+ FORMAT_PARAMETERS array
+ getFormatTemplate() string
+ getFormatParameters() array
# getFormatDefaultParameterValues() array

오류 처리

S3 Transfer Manager는 비동기 작업에 promise를 사용합니다. promise의 `otherwise` 또는 `then` 메서드를 사용하여 오류를 처리할 수 있습니다. 구현을 `try-catch` 블록으로 래핑할 수도 있습니다.

Promise 오류 처리

`otherwise` 사용:

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

`then` 사용:

```
<?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();
```

try-catch 블록 사용

```
<?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";
}
```

<S3TransferManager ### ## ### ### ## ## API ## ### #####.>

디렉터리 작업 실패 정책

디렉터리 작업의 경우 실패 정책 콜백을 지정할 수도 있습니다. 이 콜백은 개별 파일의 실패를 처리하고 디렉터리 업로드를 계속할지 여부를 결정합니다.

Example **`uploadDirectory`** 작업에 'failure_policy' 함수 사용

```
<?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 **`downloadDirectory`** 작업에 'failure_policy' 함수 사용

```
<?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();
```

고급 사용

이 섹션에서는 S3 Transfer Manager의 고급 사용 패턴 및 기술을 다룹니다.

멀티파트 업로드

S3 Transfer Manager는 대용량 파일에 멀티파트 업로드를 자동으로 사용합니다. 구성 옵션을 사용하여이 동작을 사용자 지정할 수 있습니다.

```
<?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();
```

멀티파트 다운로드

멀티파트 다운로드 동작을 사용자 지정할 수 있습니다.

```
<?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();
```

`AbstractMultipartDownloader` 클래스의 다른 멤버에 대한 자세한 내용은 API 설명서<## ## >를 참조하세요.

사용자 지정 필터

필터 함수를 사용하여 디렉터리 작업을 위한 파일을 선택적으로 업로드하거나 다운로드할 수 있습니다.

디렉터리 필터 업로드

파라미터 정보에 대한 자세한 내용은 `uploadDirectory` 메서드의 호출 가능한 필터 옵션을 참조하세요.

```
<?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();
```

디렉터리 필터 다운로드

파라미터 정보에 대한 자세한 내용은 downloadDirectory 메서드의 호출 가능한 필터 옵션을 참조하세요.

```
<?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();
```

콜백 사전 요청

사전 요청 콜백을 사용하여 각 파일의 요청 파라미터를 수정할 수 있습니다. 파라미터에 대한 자세한 내용은 다음을 참조하세요.

- `uploadDirectory` 메서드의 put_object_request_callaback 호출 가능 옵션
- `downloadDirectory` 메서드의 download_object_request_modifier 호출 가능 옵션

Example **`uploadDirectory`** 메서드에 대한 사전 요청 콜백의

```
<?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 가능 예제

이 섹션에서는 일반적인 S3 Transfer Manager 작업을 위한 정적 메서드가 포함된 전체 헬퍼 클래스를 제공합니다. 헬퍼 클래스는 사용을 간소화합니다. 내부적으로 초기화 및 구성을 처리합니다.

S3TransferHelper 클래스

```
<?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(),
        ];
    }
}
```

헬퍼 클래스 사용 예제

다음 예제에서는 S3TransferHelper 클래스를 사용하는 방법을 보여줍니다.

```
<?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";
```

AWS SDK for PHP 버전 3을 사용하여 Amazon S3에서 디렉터리 전송

AWS SDK for PHP 버전 3의 `Transfer` 클래스를 사용하여 전체 디렉터리를 Amazon S3 버킷에 업로드하고 전체 버킷을 로컬 디렉터리에 다운로드합니다.

Amazon S3에 로컬 디렉터리 업로드

`Aws\S3\Transfer` 객체가 전송을 수행합니다. 다음 예에서는 파일의 로컬 디렉터리를 Amazon S3 버킷에 재귀적으로 업로드하는 방법을 보여줍니다.

```
// 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();
```

이 예에서는 Amazon S3 클라이언트와 `Transfer` 객체를 생성하고 전송을 동기적으로 수행했습니다.

이전 예제에서는 전송을 수행하는 데 필요한 최소 코드 양을 보여줍니다. 또한 `Transfer` 객체는 전송을 비동기적으로 수행할 수 있으며 전송을 사용자 지정하는 데 사용할 수 있는 다양한 구성 옵션이 있습니다.

s3:// URI에 키 접두사를 제공하여 Amazon S3 버킷의 "하위 폴더"에 로컬 파일을 업로드할 수 있습니다. 다음 예에서는 디스크에 있는 로컬 파일을 bucket 버킷에 업로드하고 foo 키 접두사 아래에 파일을 저장합니다.

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

Amazon S3 버킷 다운로드

$source 인수를 Amazon S3 URI(예: s3://bucket)로 지정하고, $dest 인수를 로컬 디렉터리에 대한 경로로 지정하여 디스크에 있는 로컬 디렉터리에 Amazon S3 버킷을 재귀적으로 다운로드할 수 있습니다.

```
// 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**
>
> SDK는 버킷에 객체를 다운로드할 때 필요한 디렉터리를 자동으로 생성합니다.

"가상 폴더" 아래에 저장된 객체만 다운로드하려면 Amazon S3 URI에서 버킷 뒤에 키 접두사를 포함할 수 있습니다. 다음 예에서는 지정된 버킷의 "/foo" 키 접두사 아래에 저장된 파일만 다운로드합니다.

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

구성

Transfer 객체 생성자는 다음 인수를 받습니다.

$client

전송을 수행하는 데 사용할 Aws\ClientInterface 객체입니다.

$source(문자열 | **Iterator**)

전송 중인 소스 데이터입니다. 이 인수는 디스크에 있는 로컬 경로(예: /path/to/files) 또는 Amazon S3 버킷(예: s3://bucket)을 가리킬 수 있습니다. 또한 s3:// URI는 공통 접두사 아래에 있는 객체만 전송하는 데 사용 가능한 키 접두사를 포함할 수 있습니다.

$source 인수가 Amazon S3 URI인 경우 $dest 인수는 로컬 디렉터리여야 하며 그 반대의 경우도 마찬가지입니다.

문자열 값을 제공하는 이외에 절대 파일 이름을 생성하는 \Iterator 객체를 제공할 수도 있습니다. \Iterator 객체를 제공하는 경우 $options 연결 배열에 base_dir 옵션을 제공해야 합니다.

$dest

파일이 전송될 대상입니다. $source 인수가 디스크에 있는 로컬 경로인 경우 $dest는 Amazon S3 버킷 URI(예: s3://bucket)이고, $source 인수가 Amazon S3 버킷 URI인 경우 $dest 인수는 디스크에 있는 로컬 경로여야 합니다.

$options

전송 옵션의 결합형 배열입니다. 유효한 전송 옵션은 다음과 같습니다.

add_content_md5 (bool)

업로드에 대한 MD5 체크섬을 true 계산하려면 로 설정합니다.

base_dir(문자열)

$source가 반복자인 경우 소스의 기본 디렉터리입니다. $source 옵션이 배열이 아닌 경우 이 옵션이 무시됩니다.

before(callable)

각 전송 이전에 호출할 콜백입니다. 콜백에는 function (Aws\Command $command) {...}와 같은 함수 서명이 있어야 합니다. 제공되는 명령은 GetObject, PutObject, CreateMultipartUpload, UploadPart 또는 CompleteMultipartUpload 명령입니다.

mup_threshold (int)

PutObject 대신 사용할 멀티파트 업로드의 크기(바이트)입니다. 기본값은 16777216(16MB)입니다.

concurrency(정수, 기본값=5)

동시에 업로드할 파일 수입니다. 이상적인 동시성 값은 업로드 중인 파일 수와 각 파일의 평균 크기에 따라 다릅니다. 일반적으로 파일이 작을수록 동시성을 높여 이점을 얻을 수 있지만 파일이 크면 이점이 없습니다.

debug (bool)

전송에 대한 디버그 정보를 출력하려면 `true`로 설정합니다. STDOUT에 쓰지 않고 특정 스트림에 쓰려면 `fopen()` 리소스로 설정합니다.

비동기적 전송

`Transfer` 객체가 `GuzzleHttp\Promise\PromisorInterface`의 인스턴스입니다. 즉, 객체의 `promise` 메서드를 호출하여 전송을 비동기적으로 수행하고 시작할 수 있습니다.

```
$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!';
});
```

파일이 전송되지 않으면 promise가 거부됩니다. promise의 `otherwise` 메서드를 사용하여 실패한 전송을 비동기적으로 처리할 수 있습니다. `otherwise` 함수는 오류 발생 시 호출할 콜백을 받습니다. 콜백은 일반적으로의 인스턴스인 거부에 `$reason` 대해를 수락합니다`Aws\Exception\AwsException`(모든 유형의 값을 콜백에 전달할 수 있음).

```
$promise->otherwise(function ($reason) {
    echo 'Transfer failed: ';
    var_dump($reason);
});
```

`Transfer` 객체는 promise를 반환하므로 이러한 전송이 다른 비동기적 promise와 동시에 발생할 수 있습니다.

디렉터리 전송 사용자 지정

생성자에 콜백을 추가하여가 Transfer 실행하는 옵션을 사용자 지정할 수 있습니다.

```
$uploader = new Transfer($s3Client, $source, $dest, [
    'before' => function (\Aws\Command $command) {
        // Commands can vary for multipart uploads, so check which command
        // is being processed.
        if (in_array($command->getName(), ['PutObject', 'CreateMultipartUpload'])) {
            // Set custom cache-control metadata.
            $command['CacheControl'] = 'max-age=3600';
            // Apply a canned ACL.
            $command['ACL'] = strpos($command['Key'], 'CONFIDENTIAL') ### false
                ? 'public-read'
                : 'private';
        }
    },
]);
```

AWS SDK for PHP 버전 3의 Amazon S3 클라이언트 측 암호화

클라이언트 측 암호화를 사용하여 사용자 환경에서 직접 데이터를 암호화 및 암호화 해제합니다. 즉, 이 데이터를 Amazon S3에 전송하기 이전에 암호화하므로 암호화 처리를 위해 외부 서비스를 이용할 필요가 없습니다. 새로운 구현의 경우 S3EncryptionClientV3 및와 더 이상 사용되지 않는 S3EncryptionClientV2 S3EncryptionMultipartUploaderV2 및에 S3EncryptionMultipartUploaderV3 대해 S3EncryptionClient 및를 사용하는 것이 좋습니다S3EncryptionMultipartUploader. 더 이상 사용되지 않는 버전을 계속 사용하고 있는 이전 구현에서는 마이그레이션을 시도하는 것이 좋습니다. S3EncryptionClientV3는 S3EncryptionClient 레거시를 사용하여 암호화된 데이터의 암호 해독에 대한 지원을 유지합니다.

는 봉투 암호화를 AWS SDK for PHP 구현하고 암호화 및 복호화에 OpenSSL을 사용합니다. 구현은 지원하는 기능이 일치하는 다른 SDK와 상호 연동이 가능합니다. SDK의 promise 기반 비동기 워크플로와도 호환됩니다.

마이그레이션 가이드

더 이상 사용되지 않는 클라이언트에서 새 클라이언트로 마이그레이션하려는 사용자를 위해 v1에서 v2로 마이그레이션하는 마이그레이션 가이드와 v2에서 v3로 마이그레이션하는 마이그레이션 가이드가 있습니다.

설정

클라이언트 측 암호화를 시작하려면 다음이 필요합니다.

- AWS KMS 암호화 키
- S3 버킷

예제 코드를 실행하기 전에 AWS 자격 증명을 구성합니다. AWS SDK for PHP 버전 3의 자격 증명을 참조하세요.

암호화(Encryption)

에 암호화된 객체를 업로드하면 표준 파라미터 외에 4개의 추가 `PutObject` 파라미터가 `S3EncryptionClientV3` 사용됩니다.

- `'@KmsEncryptionContext'`는 암호화된 객체에 추가 보안 계층을 추가하는 데 사용할 수 있는 키-값 쌍입니다. 암호화 클라이언트는 동일한 키를 전달해야 하며, 이 키는 get 호출 시 자동으로 전달됩니다. 추가 컨텍스트가 필요하지 않은 경우 빈 배열을 전달하세요.
- `@CipherOptions`는 사용할 암호와 키 크기를 포함한 암호화를 위한 추가 구성입니다.
- `@MaterialsProvider`는 암호 키 및 초기화 벡터 생성과 암호 키 암호화를 처리할 뿐만 아니라 암호 키를 암호화는 공급자입니다.
- `@CommitmentPolicy`는 키 커밋 또는 키 커밋 없이 객체를 읽는 방법과 키 커밋 또는 키 커밋 없이 객체를 쓰는 방법을 지정하는 정책 옵션입니다.

```
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**
>
> Amazon S3 및 AWS KMS기반 서비스 오류 외에도가 올바르게 구성되지 않은 경우 `InvalidArgumentException` 객체가 발생할 수 `'@CipherOptions'` 있습니다.

해독

객체 다운로드 및 복호화에는 표준 파라미터 외에도 5개의 추가 `GetObject` 파라미터가 있으며,이 중 2개가 필요합니다. 클라이언트는 기본 암호 옵션을 자동으로 검색합니다.

- **`'@SecurityProfile'`**: 'V3'로 설정된 경우 V3-compatible으로 암호화된 객체만

  형식을 해독할 수 있습니다. 이 파라미터를 'V3_AND_LEGACY'로 설정하면 V1-compatible 형식으로 암호화된 객체도 해독할 수 있습니다. 마이그레이션을 지원하려면 @SecurityProfile을 'V3_AND_LEGACY'로 설정합니다. 새 애플리케이션 개발에만 'V3'를 사용합니다.
- **`'@MaterialsProvider'`**는 다음과 같이 암호 키 및 초기화 벡터 생성을 처리하는 공급자입니다.

  암호 키 암호화도 마찬가지입니다.
- **`'@KmsAllowDecryptWithAnyCmk'`**: (선택 사항) 이 매개변수를 true로 설정하면 복호화가 활성화됩니다.

  MaterialsProvider의 생성자에 KMS 키 ID를 제공하지 않아도 됩니다. 기본값은 false입니다.
- **`'@CipherOptions'`**(선택 사항)은 다음을 포함한 암호화를 위한 추가 구성입니다.

  사용할 암호 및 키 크기.
- **`@CommitmentPolicy`**를 사용하여 객체를 읽는 방법을 지정하는 정책 옵션

  키 커밋 또는 키 커밋 없음 및 키 커밋 또는 키 커밋 없이 객체를 작성하는 방법.

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> Amazon S3 및 AWS KMS기반 서비스 오류 외에도가 올바르게 구성되지 않은 경우 `InvalidArgumentException` 객체가 발생할 수 `'@CipherOptions'` 있습니다.

암호 구성

`'Cipher'`(문자열)

암호화 중에 암호화 클라이언트에서 사용되는 암호 메서드입니다. 현재는 'gcm'만 지원됩니다.

> **Important**
>
> PHP는 GCM 암호화를 위해 OpenSSL을 사용하여 암호화 및 해독하는 데 필요한 추가 매개변수를 포함하도록 버전 7.1로 업데이트됩니다. PHP 버전 7.0 및 이전 버전의 경우 GCM 지원을 위한 폴리필이 제공되어 암호화 클라이언트 `S3EncryptionClientV2` 및 `S3EncryptionMultipartUploaderV2`에서 사용됩니다. 그러나 대규모 입력의 경우 폴리필을 사용하면 PHP 7.1+의 기본 구현을 사용하는 것보다 성능이 훨씬 느려지므로 효과적으로 사용하려면 이전 PHP 버전 환경을 업그레이드해야 할 수 있습니다.

`'KeySize'` (int)

암호화를 위해 생성할 콘텐츠 암호화 키의 길이입니다. 기본값은 256비트입니다. 유효한 구성 옵션은 256비트입니다.

`'Aad'`(문자열)

암호화된 페이로드와 함께 포함할 선택적 '추가 인증 데이터'입니다. 이 정보는 암호화를 풀 때 확인됩니다. `Aad`는 'gcm' 암호를 사용할 경우에만 사용할 수 있습니다.

> **Important**
>
> 추가 인증 데이터는 모든 AWS SDKs에서 지원되지 않으므로 다른 SDKs는이 파라미터를 사용하여 암호화된 파일을 해독하지 못할 수 있습니다.

메타데이터 전략

`Aws\Crypto\MetadataStrategyInterface`를 구현하는 클래스의 인스턴스를 제공할 수도 있습니다. 이 간단한 인터페이스는 봉투 암호화 자료를 포함하는 `Aws\Crypto\MetadataEnvelope`의 저장 및 로드를 처리합니다. SDK는 이를 구현하는 `Aws\S3\Crypto\HeadersMetadataStrategy` 및 `Aws\S3\Crypto\InstructionFileMetadataStrategy` 클래스를 제공합니다. 기본적으로 `HeadersMetadataStrategy`가 사용됩니다.

```
$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,
]);
```

HeadersMetadataStrategy 및 InstructionFileMetadataStrategy의 클래스 이름 상수는 ::class 를 호출하여 제공할 수도 있습니다.

```
$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**
>
> 지침 파일을 업로드한 이후에 오류가 발생하는 경우 지침 파일이 자동으로 삭제됩니다.

멀티파트 업로드

클라이언트 측 암호화를 사용하여 멀티파트 업로드를 수행할 수도 있습니다. `Aws\S3\Crypto\S3EncryptionMultipartUploaderV3`는 업로드 전에 암호화를 위해 소스 스트림을 준비합니다. `Aws\S3\MultipartUploader` 및 `Aws\S3\Crypto\S3EncryptionClientV3`를 사용할 때와 비슷한 방식으로 생성합니다. `S3EncryptionMultipartUploaderV3`는 `'@MetadataStrategy'`와 동일한 `S3EncryptionClientV3` 옵션과 모든 사용 가능한 `'@CipherOptions'` 구성을 처리할 수 있습니다.

```
$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**
>
> Amazon S3 및 AWS KMS기반 서비스 오류 외에도가 올바르게 구성되지 않은 경우 `InvalidArgumentException` 객체가 발생할 수 `'@CipherOptions'` 있습니다.

체크섬을 통한 데이터 무결성 보호

Amazon Simple Storage Service(S3)는 객체를 업로드할 때 체크섬을 지정하는 기능을 제공합니다. 체크섬을 지정하면 객체와 함께 저장되며 객체를 다운로드할 때 유효성을 검사할 수 있습니다.

체크섬은 파일을 전송할 때 데이터 무결성을 한층 더 강화합니다. 체크섬을 사용하면 수신된 파일이 원본 파일과 일치하는지 확인하여 데이터 일관성을 확인할 수 있습니다. Amazon S3의 체크섬에 대한 자세한 내용은 지원되는 알고리즘을 포함한 Amazon Simple Storage Service 사용 설명서를 참조하세요.

필요에 가장 적합한 알고리즘을 유연하게 선택하고 SDK가 체크섬을 계산하도록 할 수 있습니다. 또는 지원되는 알고리즘 중 하나를 사용하여 미리 계산된 체크섬 값을 제공할 수 있습니다.

> **Note**
>
> 버전 3.337.0부터 AWS SDK for PHP SDK는 업로드를 위한 CRC32 체크섬을 자동으로 계산하여 기본 무결성 보호를 제공합니다. 사전 계산된 체크섬 값을 제공하지 않거나 SDK가 체크섬을 계산하는 데 사용해야 하는 알고리즘을 지정하지 않은 경우 SDK는이 체크섬을 계산합니다. 또한 SDK는 AWS SDK 및 도구 참조 안내서에서 확인할 수 있고 외부에서 설정할 수 있는 데이터 무결성 보호에 대한 전역 설정을 지원합니다.

> **Important**
>
> CRC32C 알고리즘을 사용하려면 PHP 환경에AWS 공통 런타임(AWS CRT) 확장을 설치해야 합니다.

체크섬은 객체 업로드와 객체 다운로드라는 두 가지 요청 단계로 설명합니다.

객체 업로드

`S3Client`의 PutObject 메서드를 사용하여 Amazon S3에 객체를 업로드합니다. 파라미터 배열의 `ChecksumAlgorithm` 쌍을 사용하여 체크섬 계산을 활성화하고 알고리즘을 지정합니다.

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

요청에 체크섬 알고리즘을 제공하지 않는 경우 체크섬 동작은 다음 표와 같이 사용하는 SDK 버전에 따라 달라집니다.

체크섬 알고리즘이 제공되지 않은 경우 체크섬 동작

PHP SDK 버전	체크섬 동작
3.337.0 이하	SDK는 CRC 기반 체크섬을 자동으로 계산하여 요청에 제공하지 않습니다.
3.337.0 이상	SDK는 CRC32 알고리즘을 사용하여 체크섬을 계산하고 요청에 제공합니다. Amazon S3는 자체 CRC32 체크섬을 계산하여 전송의 무결성을 검증하고 이를 SDK에서 제공하는 체크섬과 비교합니다. 체크섬이 일치하면 체크섬이 객체와 함께 저장됩니다.

미리 계산된 체크섬 값 사용

요청과 함께 제공되는 사전 계산된 체크섬 값은 SDK의 자동 계산을 비활성화하고 제공된 값을 대신 사용합니다.

다음 예제에서는 사전 계산된 SHA256 체크섬이 있는 요청을 보여줍니다.

```
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
]);
```

Amazon S3에서 체크섬 값이 지정된 알고리즘에 대해 올바르지 않다고 판단하면 서비스는 오류 응답을 반환합니다.

멀티파트 업로드

멀티파트 업로드에 체크섬을 사용할 수도 있습니다.

다음 예제에서 볼 수 있듯이 MultipartUploader 생성자의 params 배열에서 체크섬 알고리즘을 키-값 쌍으로 지정합니다.

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

객체 다운로드

getObject 메서드를 사용하여 객체를 다운로드하면, ChecksumMode 키 값이 enabled인 경우 SDK가 자동으로 체크섬을 확인합니다.

다음 스니펫의 요청은 체크섬을 계산하고 값을 비교하여 응답의 체크섬을 검증하도록 SDK에 지시합니다.

```
$result = $client->getObject([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'test-checksum-key',
        'ChecksumMode' => 'enabled',
]);
```

> **Note**
>
> 체크섬과 함께 객체를 업로드하지 않은 경우 검증이 수행되지 않습니다.

AWS SDK for PHP에 대한 지침이 포함된 코드 예제

이 섹션에는 AWS SDK for PHP를 사용하는 일반적인 AWS 시나리오를 보여주는 코드 예제가 포함되어 있습니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용한 Amazon CloudFront 예제
- AWS SDK for PHP 버전 3을 사용한 사용자 지정 Amazon CloudSearch 도메인 요청에 서명
- AWS SDK for PHP 버전 3을 사용한 Amazon CloudWatch 예제
- AWS SDK for PHP 버전 3을 사용한 Amazon EC2 예제
- AWS SDK for PHP 버전 3으로 Amazon OpenSearch Service 검색 요청에 서명
- AWS SDK for PHP 버전 3을 사용한 AWS Identity and Access Management 예제

- AWS SDK for PHP 버전 3을 사용한 AWS Key Management Service 예제
- AWS SDK for PHP 버전 3을 사용한 Amazon Kinesis 예제
- AWS SDK for PHP 버전 3을 사용한 AWS Elemental MediaConvert 예제
- AWS SDK for PHP 버전 3을 사용한 Amazon S3 예제
- Secrets Manager API 및 AWS SDK for PHP 버전 3을 사용한 암호 관리
- AWS SDK for PHP 버전 3을 사용한 Amazon SES 예제
- AWS SDK for PHP 버전 3을 사용한 Amazon SNS 예제
- AWS SDK for PHP 버전 3을 사용한 Amazon SQS 예제
- AWS SDK for PHP 버전 3을 사용하여 Amazon EventBridge 글로벌 엔드포인트로 이벤트 전송

AWS SDK for PHP 버전 3을 사용한 Amazon CloudFront 예제

Amazon CloudFront는 자체 웹 서버 또는 Amazon S3와 같은 AWS 서버에서 정적 및 동적 웹 콘텐츠를 더 빠르게 제공하는 AWS 웹 서비스입니다. CloudFront는 엣지 로케이션이라고 하는 데이터 센터의 전 세계 네트워크를 통해 콘텐츠를 제공합니다. CloudFront를 통해 배포하는 콘텐츠를 사용자가 요청하면 지연 시간이 가장 낮은 엣지 로케이션으로 라우팅됩니다. 콘텐츠가 아직 캐싱되지 않은 경우에는 CloudFront가 오리진 서버에서 복사본을 가져와 공급한 후 향후 요청 시 사용할 수 있도록 캐싱합니다.

CloudFront에 대한 자세한 내용은 Amazon CloudFront 개발자 가이드를 참조하세요.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

CloudFront API 및 AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront 배포 관리

Amazon CloudFront는 웹 서버 또는 Amazon 서비스(예: Amazon S3 및 Amazon EC2)에 저장되어 있는 정적/동적 파일을 더욱 빠르게 배포할 목적으로 콘텐츠를 전 세계 엣지 로케이션에서 캐싱합니다. 이후 사용자가 웹사이트에서 콘텐츠를 요청하면 CloudFront가 파일이 캐싱되어 있다는 가정 하에 가장 가까운 엣지 로케이션에서 해당 콘텐츠를 공급합니다. 그렇지 않으면 CloudFront가 파일 복사본을 가져와 공급한 후 다음 요청 시 사용할 수 있도록 캐싱합니다. 엣지 로케이션에서 콘텐츠를 캐싱하면 해당 지역에서 유사한 사용자 요청에 따른 지연 시간을 줄일 수 있습니다.

CloudFront 배포를 생성할 때는 항상 콘텐츠의 위치와 사용자 요청 시 배포 방법을 지정합니다. 이번 주제에서는 HTML, CSS, JSON 및 이미지 파일 같은 정적/동적 파일의 배포를 중심으로 설명하겠습니다. CloudFront를 비디오 온디맨드(VOD)에 사용하는 방법에 대한 자세한 내용은 CloudFront를 사용하는 온디맨드 및 라이브 스트리밍 비디오 단원을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateDistribution을 사용하여 배포를 생성합니다.
- GetDistribution을 사용하여 배포를 가져옵니다.
- ListDistributions를 사용하여 배포를 나열합니다.
- UpdateDistributions를 사용하여 배포를 업데이트합니다.
- DisableDistribution을 사용하여 배포를 비활성화합니다.
- DeleteDistributions를 사용하여 배포를 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하세요.

CloudFront 배포 생성

Amazon S3 버킷에서 배포를 생성합니다. 다음 예제에서는 선택 사항인 파라미터를 주석 처리하였지만 기본값은 표시하였습니다. 사용자 지정을 배포에 추가하려면 기본값과 파라미터를 `$distribution`에서 주석 해제하세요.

CloudFront 배포를 생성할 때는 CreateDistribution 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

CloudFront 배포 가져오기

지정한 CloudFront 배포의 상태와 세부 정보를 가져올 때는 GetDistribution 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
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 배포 목록

현재 계정에서 지정한 AWS 리전에서 기존 CloudFront 배포 목록을 가져올 때는 ListDistributions 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
function listDistributions($cloudFrontClient)
{
    try {
        $result = $cloudFrontClient->listDistributions([]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}
```

```
function listTheDistributions()
{
    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-2'
    ]);

    $distributions = listDistributions($cloudFrontClient);

    if (count($distributions) == 0) {
        echo 'Could not find any distributions.';
    } else {
        foreach ($distributions['DistributionList']['Items'] as $distribution) {
            echo 'The distribution with the ID of ' . $distribution['Id'] .
                ' has the status of ' . $distribution['Status'] . '.' . "\n";
        }
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheDistributions();
```

CloudFront 배포 업데이트

CloudFront 배포를 업데이트하는 방법은 배포를 생성하는 방법과 비슷합니다. 단, 배포를 업데이트할 때는 필드가 더 많이 필요하고, 모든 값이 포함되어야 합니다. 기존 배포를 변경하려면 먼저 기존 배포를 가져온 다음 `$distribution` 배열에서 변경할 값을 업데이트하는 것이 좋습니다.

지정한 CloudFront 배포를 업데이트할 때는 UpdateDistribution 작업을 사용합니다.

가져옵니다.

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function updateDistribution(
```

```
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);

        return 'The distribution with the following effective URI has ' .
            'been updated: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}
```

```
function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function updateADistribution()
{
    // $distributionId = 'E1BTGP2EXAMPLE';
    $distributionId = 'E1X3BKQ569KEMH';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To change a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }
```

```
    // To change a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration.
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    // To change a distribution's configuration, you can set the
    // distribution's related configuration value as part of a change request,
    // for example:
    // 'Enabled' => true
    // Some configuration values are required to be specified as part of a change
    // request, even if you don't plan to change their values. For ones you
    // don't want to change but are required to be specified, you can just reuse
    // their current values, as follows.
    $distributionConfig = [
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => $currentConfig['DistributionConfig']["Enabled"],
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo updateDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
```

```
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateADistribution();
```

CloudFront 배포 사용 중지

배포를 비활성화하거나 제거하려면 상태를 deployed에서 disabled로 변경합니다.

지정한 CloudFront 배포를 비활성화할 때는 DisableDistribution 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
function disableDistribution(
    $cloudFrontClient,
    $distributionId,
    $distributionConfig,
    $eTag
) {
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution with the following effective URI has ' .
            'been disabled: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
```

```
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig'])) {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
```

```
    }
}

function disableADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To disable a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To delete a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration, including setting the new
    // configuration to "disabled".
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    $distributionConfig = [
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'CallerReference' => $currentConfig['DistributionConfig']["CallerReference"],
        'Comment' => $currentConfig['DistributionConfig']["Comment"],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']
["DefaultRootObject"],
        'Enabled' => false,
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
```

```
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
        'PriceClass' => $currentConfig['DistributionConfig']["PriceClass"],
        'Restrictions' => $currentConfig['DistributionConfig']["Restrictions"],
        'ViewerCertificate' => $currentConfig['DistributionConfig']
["ViewerCertificate"],
        'WebACLId' => $currentConfig['DistributionConfig']["WebACLId"]
    ];

    echo disableDistribution(
        $cloudFrontClient,
        $distributionId,
        $distributionConfig,
        $eTag['ETag']
    );
}

// Uncomment the following line to run this code in an AWS account.
// disableADistribution();
```

CloudFront 배포 삭제

배포가 disabled 상태일 때는 해당 배포를 삭제할 수 있습니다.

지정한 CloudFront 배포를 삭제할 때는 DeleteDistribution 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
function deleteDistribution($cloudFrontClient, $distributionId, $eTag)
{
    try {
        $result = $cloudFrontClient->deleteDistribution([
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution at the following effective URI has ' .
```

```
            'been deleted: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag'])) {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function deleteADistribution()
{
    $distributionId = 'E17G7YNEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To delete a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);
```

```
    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    } else {
        echo deleteDistribution(
            $cloudFrontClient,
            $distributionId,
            $eTag['ETag']
        );
    }
}

// Uncomment the following line to run this code in an AWS account.
// deleteADistribution();
```

CloudFront API 및 AWS SDK for PHP 버전 3을 사용한 Amazon CloudFront 무효화 관리

Amazon CloudFront는 전 세계 엣지 로케이션에 정적/동적 파일의 복사본을 캐싱합니다. 모든 엣지 로케이션에서 파일을 제거하거나 업데이트하려면 각 파일 또는 파일 그룹마다 무효화를 생성합니다.

매월 첫 무효화 1,000개는 무료입니다. CloudFront 엣지 로케이션에서 콘텐츠를 제거하는 방법에 대한 자세한 내용은 파일 무효화 단원을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateInvalidation을 사용하여 배포 무효화를 생성합니다.
- GetInvalidation을 사용하여 배포 무효화를 가져옵니다.
- ListInvalidations를 사용하여 배포 무효화를 나열합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하세요.

배포 무효화 생성

CloudFront 배포 무효화는 제거할 파일의 경로 위치를 지정하여 생성합니다. 이번 예제에서는 배포에 속한 파일을 모두 무효화하지만 원한다면 `Items`에서 특정 파일을 지정할 수 있습니다.

CloudFront 배포 무효화를 생성할 때는 CreateInvalidation 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

배포 무효화 가져오기

CloudFront 배포 무효화에 대한 상태와 세부 정보를 가져올 때는 GetInvalidation 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

배포 무효화 나열

현재 CloudFront 배포 무효화를 모두 나열할 때는 ListInvalidations 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
function listInvalidations($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->listInvalidations([
            'DistributionId' => $distributionId
        ]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheInvalidations()
{
    $distributionId = 'E1WICG1EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $invalidations = listInvalidations(
```

```
        $cloudFrontClient,
        $distributionId
    );

    if (isset($invalidations['InvalidationList'])) {
        if ($invalidations['InvalidationList']['Quantity'] > 0) {
            foreach ($invalidations['InvalidationList']['Items'] as $invalidation) {
                echo 'The invalidation with the ID of ' . $invalidation['Id'] .
                    ' has the status of ' . $invalidation['Status'] . '.' . "\n";
            }
        } else {
            echo 'Could not find any invalidations for the specified distribution.';
        }
    } else {
        echo 'Error: Could not get invalidation information. Could not get ' .
            'information about the specified distribution.';
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheInvalidations();
```

AWS SDK for PHP 버전 3으로 Amazon CloudFront URL에 서명

서명된 URL을 사용하여 사용자에게 프라이빗 콘텐츠에 대한 액세스를 제공할 수 있습니다. 서명된 URL에는 만료 날짜 같은 추가 정보가 포함되므로 콘텐츠에 대한 액세스를 더욱 세부적으로 제어할 수 있습니다. 이러한 추가 정보는 미리 준비된(canned) 정책 또는 사용자 지정 정책에 따라 정책 설명에 나타납니다. 프라이빗 배포를 설정하는 방법과 URL에 서명해야 하는 이유에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 Amazon CloudFront를 통한 프라이빗 콘텐츠 제공을 참조하세요.

- getSignedURL을 사용하여 서명된 Amazon CloudFront URL을 생성합니다.
- getSignedCookie를 사용하여 서명된 Amazon CloudFront 쿠키를 생성합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하세요.

프라이빗 배포를 위한 CloudFront URL 서명

SDK에서 CloudFront 클라이언트를 사용하여 URL에 서명할 수 있습니다. 먼저 `CloudFrontClient` 객체를 생성해야 합니다. 미리 준비된 정책 또는 사용자 지정 정책을 사용하여 비디오 리소스에 대한 CloudFront URL에 서명할 수 있습니다.

가져옵니다.

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function signPrivateDistribution(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistribution()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
```

```
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistribution(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistribution();
```

CloudFront URL 생성 시 사용자 지정 정책 사용

사용자 지정 정책을 사용하려면 `policy` 대신 `expires` 키를 제공합니다.

가져옵니다.

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

CloudFront에서 서명된 URL 사용

서명된 URL의 형태는 서명하는 URL이 “HTTP” 체계를 사용하는지 또는 "RTMP” 체계를 사용하는지에 따라 다릅니다. “HTTP”의 경우 전체 절대 URL이 반환됩니다. “RTMP”의 경우 편의를 위해 상대 URL만 반환됩니다. 이렇게 하는 이유는 일부 플레이어에서 호스트와 경로를 별도의 파라미터로 제공해야 하기 때문입니다.

다음 예제에서는 JWPlayer를 사용하여 비디오를 표시하는 웹 페이지를 생성하기 위해 서명된 URL을 사용하는 방법을 보여 줍니다. FlowPlayer와 같은 다른 플레이어에 동일한 유형이 기술이 적용되지만, 다른 클라이언트 측 코드가 필요합니다.

```
<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>
```

프라이빗 배포를 위한 CloudFront 쿠키 서명

서명된 URL에 대한 대체 방법으로, 서명된 쿠키를 통해 프라이빗 배포에 대한 클라이언트 액세스 권한을 부여할 수도 있습니다. 서명된 쿠키를 사용하면 HLS 형식의 비디오용 모든 파일 또는 웹 사이트의 구독자 영역에 있는 모든 파일과 같은 여러 개의 제한된 파일에 대한 액세스 권한을 제공할 수 있습니

다. 서명된 URL 대신 서명된 쿠키를 사용해야 하는 이유(또는 그 반대)에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 서명된 URL과 서명된 쿠키 중 선택을 참조하세요.

서명된 쿠키를 생성하는 방법은 서명된 URL을 생성하는 방법과 비슷합니다. 유일한 차이점은 호출되는 메서드입니다(`getSignedCookie` 대신 `getSignedUrl`).

가져옵니다.

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function signCookie(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}

function signACookie()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';
```

```
    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookie(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Expires = 1589926678
    CloudFront-Signature = Lv1DyC2q...2HPXaQ__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookie();
```

CloudFront 쿠키 생성 시 사용자 지정 정책 사용

getSignedUrl과 마찬가지로, 'policy' 파라미터 및 expires 파라미터 대신 url 파라미터를 제공하여 사용자 지정 정책으로 쿠키에 서명할 수 있습니다. 사용자 지정 정책에서는 리소스 키에 와일드카드가 포함될 수 있습니다. 이렇게 하면 여러 개의 파일에 사용할 하나의 서명된 쿠키를 생성할 수 있습니다.

getSignedCookie는 프라이빗 배포에 대한 액세스 권한을 부여하기 위해 모두 쿠키로 설정해야 하는 키-값 페어의 배열을 반환합니다.

가져옵니다.

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

use Aws\CloudFront\CloudFrontClient;
```

```
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

CloudFront 쿠키를 Guzzle 클라이언트에게 전송

이러한 쿠키를 Guzzle 클라이언트에 사용하기 위해 `GuzzleHttp\Cookie\CookieJar`에 전달할 수도 있습니다.

```
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');
```

자세한 내용을 알아보려면 Amazon CloudFront 개발자 안내서의 서명된 쿠키 사용을 참조하세요.

AWS SDK for PHP 버전 3을 사용한 사용자 지정 Amazon CloudSearch 도메인 요청에 서명

AWS SDK for PHP가 지원하는 범위를 초과하여 Amazon CloudSearch 도메인 요청을 사용자 지정할 수 있습니다. IAM 인증으로 보호되는 도메인에 사용자 지정 요청을 수행해야 하는 경우 SDK의 보안 인증 공급자 및 서명자를 사용하여 모든 PSR-7 요청에 서명할 수 있습니다.

예를 들어, Cloud Search 시작 안내서를 따라 IAM으로 보호되는 도메인을 3단계에 사용하려는 경우 다음과 같이 요청에 서명하고 요청을 실행해야 합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- SignatureV4를 사용해 AWS 서명 프로토콜에 따라 요청에 서명합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon CloudSearch 도메인 요청에 서명

가져옵니다.

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

use Aws\Credentials\CredentialProvider;
use Aws\Signature\SignatureV4;
use GuzzleHttp\Client;
use GuzzleHttp\Psr7\Request;
```

샘플 코드

```
function searchDomain(
    $client,
    $domainName,
    $domainId,
```

```
    $domainRegion,
    $searchString
) {
    $domainPrefix = 'search-';
    $cloudSearchDomain = 'cloudsearch.amazonaws.com';
    $cloudSearchVersion = '2013-01-01';
    $searchPrefix = 'search?';

    // Specify the search to send.
    $request = new Request(
        'GET',
        "https://$domainPrefix$domainName-$domainId.$domainRegion." .
            "$cloudSearchDomain/$cloudSearchVersion/" .
            "$searchPrefix$searchString"
    );

    // Get default AWS account access credentials.
    $credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

    // Sign the search request with the credentials.
    $signer = new SignatureV4('cloudsearch', $domainRegion);
    $request = $signer->signRequest($request, $credentials);

    // Send the signed search request.
    $response = $client->send($request);

    // Report the search results, if any.
    $results = json_decode($response->getBody());

    $message = '';

    if ($results->hits->found > 0) {
        $message .= 'Search results:' . "\n";

        foreach ($results->hits->hit as $hit) {
            $message .= $hit->fields->title . "\n";
        }
    } else {
        $message .= 'No search results.';
    }

    return $message;
}
```

```
function searchADomain()
{
    $domainName = 'my-search-domain';
    $domainId = '7kbitd6nyiglhdtmssxEXAMPLE';
    $domainRegion = 'us-east-1';
    $searchString = 'q=star+wars&return=title';
    $client = new Client();

    echo searchDomain(
        $client,
        $domainName,
        $domainId,
        $domainRegion,
        $searchString
    );
}

// Uncomment the following line to run this code in an AWS account.
// searchADomain();
```

AWS SDK for PHP 버전 3을 사용한 Amazon CloudWatch 예제

Amazon CloudWatch (CloudWatch)는 Amazon Web Services 리소스 및 AWS에서 실행되는 애플리케이션을 실시간으로 모니터링합니다. CloudWatch를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다. CloudWatch 경보는 알림을 보내거나 정의한 규칙을 기준으로 모니터링하는 리소스를 자동으로 변경합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch 경보 작업
- AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch에서 지표 가져오기
- AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch에서 사용자 지정 지표 게시
- AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch Events에 이벤트 전송
- AWS SDK for PHP 버전 3를 사용하여 Amazon CloudWatch 경보에서 경보 작업 사용

AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch 경보 작업

Amazon CloudWatch 경보는 지정한 기간 동안 단일 지표를 감시합니다. 기간 수에 대한 주어진 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeAlarms를 사용하여 경보에 대해 설명합니다.
- PutMetricAlarm을 사용하여 경보를 생성합니다.
- DeleteAlarms를 사용하여 경보를 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

경보 설명

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

경보 만들기

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

경보 삭제

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function deleteAlarms($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->deleteAlarms([
            'AlarmNames' => $alarmNames
        ]);

        return 'The specified alarms at the following effective URI have ' .
```

```
            'been deleted or do not currently exist: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function deleteTheAlarms()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo deleteAlarms($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// deleteTheAlarms();
```

AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch에서 지표 가져오기

지표는 시스템 성능에 대한 데이터입니다. Amazon EC2 인스턴스 같은 일부 리소스나 자체 애플리케이션 지표에 대한 세부 모니터링을 활성화할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListMetrics를 사용하여 지표를 나열합니다.
- DescribeAlarmsForMetric을 사용하여 지표에 대한 경보를 검색합니다.
- GetMetricStatistics를 사용하여 지정된 지표에 대한 통계를 가져옵니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

지표 나열

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

지표에 대한 경보 검색

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

지표 통계 가져오기

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch에서 사용자 지정 지표 게시

지표는 시스템 성능에 대한 데이터입니다. 경보는 지정한 기간 동안 단일 지표를 감시합니다. 경보는 기간 수에 대해 주어진 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- PutMetricData를 사용하여 지표 데이터를 게시합니다.
- PutMetricAlarm을 사용하여 경보를 생성합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

지표 데이터 게시

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

경보 만들기

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
```

```
    $comparison,
    $threshold,
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
```

```
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
        $statistic,
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

AWS SDK for PHP 버전 3을 사용하여 Amazon CloudWatch Events에 이벤트 전송

CloudWatch Events는 다양한 대상으로의 Amazon Web Services(AWS) 리소스 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 전달합니다. 단순한 규칙을 사용하여 이벤트를 하나 이상의 대상 함수 또는 스트림에 일치시키고 라우팅할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- PutRule을 사용하여 규칙을 생성합니다.
- PutTargets을 사용하여 규칙에 대상을 추가합니다.
- PutEvents를 사용하여 CloudWatch 이벤트에 사용자 지정 이벤트를 전송할 수 있습니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

규칙 생성

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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());
}
```

규칙에 대상 추가

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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());
}
```

사용자 지정 이벤트 전송

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putEvents([
        'Entries' => [ // REQUIRED
            [
                'Detail' => '<string>',
                'DetailType' => '<string>',
                'Resources' => ['<string>'],
                'Source' => '<string>',
                'Time' => time()
            ],
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3를 사용하여 Amazon CloudWatch 경보에서 경보 작업 사용

경보 작업을 사용하여 Amazon EC2 인스턴스를 자동으로 중지, 종료, 재부팅 또는 복구하는 경보를 생성합니다. 인스턴스를 더는 실행할 필요가 없을 때 중지 또는 종료 작업을 사용할 수 있습니다. 재부팅 및 복구 작업을 사용하여 인스턴스를 자동으로 재부팅할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- EnableAlarmActions를 사용하여 지정된 경보에 대한 작업을 활성화합니다.
- DisableAlarmActions를 사용하여 지정된 경보에 대한 작업을 비활성화합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

경보 작업 활성화

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
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();
```

경보 작업 비활성화

가져옵니다.

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
function disableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->disableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been disabled.';
        } else {
            return 'Actions for some matching alarms ' .
```

```
                'might not have been disabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function disableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo disableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// disableTheAlarmActions();
```

AWS SDK for PHP 버전 3을 사용한 Amazon EC2 예제

Amazon Elastic Compute Cloud(Amazon EC2)는 클라우드에서 가상 서버 호스팅을 제공하는 웹 서비스입니다. 이 서비스는 크기 조정 가능한 컴퓨팅 파워를 제공하여 개발자가 더 쉽게 웹 규모 조정 클라우드 컴퓨팅을 수행할 수 있도록 설계되었습니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용하여 Amazon EC2 인스턴스 관리
- AWS SDK for PHP 버전 3를 사용하여 Amazon EC2로 탄력적 IP 주소 사용
- AWS SDK for PHP 버전 3를 사용하여 Amazon EC2의 지역 및 가용 영역 사용
- AWS SDK for PHP 버전 3을 사용하여 Amazon EC2 키 페어로 작업하기

- AWS SDK for PHP 버전 3를 사용하여 Amazon EC2 에서 보안 그룹과 작업하기

AWS SDK for PHP 버전 3을 사용하여 Amazon EC2 인스턴스 관리

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- 설명 인스턴스를 사용하여 Amazon EC2 인스턴스 설명.
- MonitorInstances를 사용하여 실행 중인 인스턴스에 대한 세부 모니터링을 활성화합니다.
- UnmonitorInstances를 사용하여 실행 중인 인스턴스에 대한 모니터링을 비활성화합니다.
- StartInstances를 사용하여 이전에 중지한 Amazon EBS 지원 AMI를 시작합니다.
- StopInstances를 사용하여 Amazon EBS 지원 인스턴스를 중지합니다.
- RebootInstances를 사용하여 하나 이상의 인스턴스 재부팅을 요청합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

인스턴스 설명

가져옵니다.

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

use Aws\Ec2\Ec2Client;
```

샘플 코드

```
$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";
    }
}
```

모니터링 활성화 및 비활성화

가져옵니다.

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

샘플 코드

```
$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);
```

인스턴스 시작 및 중지

가져옵니다.

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

샘플 코드

```
$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);
```

인스턴스 재부팅

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
```

```
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

$result = $ec2Client->rebootInstances([
    'InstanceIds' => $instanceIds
]);

var_dump($result);
```

AWS SDK for PHP 버전 3를 사용하여 Amazon EC2로 탄력적 IP 주소 사용

탄력적 IP 주소는 동적 클라우드 컴퓨팅을 위해 고안된 고정 IP 주소입니다. 탄력적 IP 주소는 AWS 계정 계정과 연결됩니다. 퍼블릭 IP 주소로 인터넷에서 연결할 수 있습니다. 인스턴스에 퍼블릭 IP 주소가 없는 경우 탄력적 IP 주소를 인스턴스에 연결하여 인터넷과의 통신을 활성화할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeInstances를 사용하여 하나 이상의 인스턴스를 설명합니다.
- AllocateAddress를 사용하여 탄력적 IP 주소를 획득합니다.
- AssociateAddress를 사용하여 탄력적 IP 주소를 인스턴스와 연결합니다.
- ReleaseAddress를 사용하여 탄력적 IP 주소를 해제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

인스턴스 설명

가져옵니다.

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

use Aws\Ec2\Ec2Client;
```

샘플 코드

```
$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";
    }
}
```

주소 할당 및 연결

가져옵니다.

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

샘플 코드

```
$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);
```

주소 릴리스

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$associationID = 'AssociationID';

$allocationID = 'AllocationID';

$result = $ec2Client->disassociateAddress([
    'AssociationId' => $associationID,
]);

$result = $ec2Client->releaseAddress([
    'AllocationId' => $allocationID,
]);

var_dump($result);
```

AWS SDK for PHP 버전 3를 사용하여 Amazon EC2의 지역 및 가용 영역 사용

Amazon EC2는 전 세계의 여러 곳에서 호스팅되고 있습니다. 해당 위치는 AWS 리전 및 가용 영역으로 구성됩니다. 각 리전은 지리적 개별 영역이며, 가용 영역이라고 알려진 여러 개의 격리된 위치가 있습니다. Amazon EC2는 인스턴스와 데이터를 여러 위치에 배치할 수 있는 기능을 제공합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeAvailabilityZones를 사용하여 사용 가능한 가용 영역을 설명합니다.
- DescribeRegions를 사용하여 현재 사용 가능한 AWS 리전을 설명합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

가용 영역 설명

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

리전 설명

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

AWS SDK for PHP 버전 3을 사용하여 Amazon EC2 키 페어로 작업하기

Amazon EC2는 퍼블릭 키 암호화 기법을 사용하여 로그인 정보를 암호화 및 해독합니다. 퍼블릭 키 암호화에서는 퍼블릭 키를 사용하여 데이터를 암호화합니다. 그런 다음 수신자가 프라이빗 키로 그 데이터를 해독합니다. 퍼블릭 키와 프라이빗 키를 키 페어라고 합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateKeyPair를 사용하여 2048비트 RSA 키 페어를 생성합니다.
- DeleteKeyPair를 사용하여 지정된 키 페어를 삭제합니다.
- DescribeKeyPairs를 사용하여 하나 이상의 키 페어를 설명합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

키 페어 생성

가져옵니다.

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

샘플 코드

```
$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);
```

키 페어 삭제

가져옵니다.

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

샘플 코드

```
$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);
```

키 페어 설명

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

AWS SDK for PHP 버전 3를 사용하여 Amazon EC2 에서 보안 그룹과 작업하기

Amazon EC2 보안 그룹은 하나 이상의 인스턴스에 대한 트래픽을 제어하는 가상 방화벽 역할을 합니다. 연결된 인스턴스에서 트래픽을 주고 받을 수 있도록 각 보안 그룹에 규칙을 추가합니다. 언제든지 보안 그룹에 대한 규칙을 수정할 수 있습니다. 새 규칙은 보안 그룹에 연결된 모든 인스턴스에 자동으로 적용됩니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeSecurityGroups를 사용하여 하나 이상의 보안 그룹을 설명합니다.
- AuthorizeSecurityGroupIngress를 사용하여 수신 규칙을 보안 그룹에 추가합니다.
- CreateSecurityGroup을 사용하여 보안 그룹을 생성합니다.
- DeleteSecurityGroup을 사용하여 보안 그룹을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

보안 그룹 설명

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

수신 규칙 추가

가져옵니다.

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

샘플 코드

```
$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);
```

보안 그룹 생성

가져옵니다.

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

샘플 코드

```
$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';
```

보안 그룹 삭제

가져옵니다.

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

샘플 코드

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$securityGroupId = 'my-security-group-id';

$result = $ec2Client->deleteSecurityGroup([
    'GroupId' => $securityGroupId
]);

var_dump($result);
```

AWS SDK for PHP 버전 3으로 Amazon OpenSearch Service 검색 요청에 서명

Amazon OpenSearch Service는 인기 있는 오픈 소스 검색 및 분석 엔진인 Amazon OpenSearch Service를 손쉽게 배포, 운영, 확장할 수 있는 관리형 서비스입니다. OpenSearch Service를 사용하면 Amazon OpenSearch Service로 직접 액세스할 수 있습니다. 따라서 개발자는 친숙한 도구뿐 아니라 확실한 보안 옵션을 사용할 수 있습니다. 많은 Amazon OpenSearch Service 클라이언트는 요청 서명을 지원하지만, 지원하지 않는 클라이언트를 사용하는 경우 AWS SDK for PHP의 내장 보안 인증 공급자 및 서명자를 사용하여 임의 PSR-7 요청에 서명할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- SignatureV4를 사용해 AWS 서명 프로토콜에 따라 요청에 서명합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

OpenSearch Service 요청 서명

OpenSearch 서비스는 Signature 버전 4를 사용합니다. 따라서 서비스의 서명 이름(이 경우 `es`)과 OpenSearch Service 도메인의 AWS 리전에 대해 요청에 서명해야 합니다. OpenSearch 서비스가 지원하는 지역의 전체 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트 페이지에서 찾을 수 있습니다. 하지만 이번 예제에서는 `us-west-2` 리전의 OpenSearch Service 도메인에 대해 요청에 서명합니다.

보안 인증을 제공해야 하며, SDK의 기본 공급자 체인 또는 AWS SDK for PHP 버전 3 보안 인증에 설명된 모든 형식의 보안 인증을 사용하여 이렇게 할 수 있습니다. PSR-7 요청도 필요합니다(아래 코드에서 `$psr7Request`라는 이름으로 가정됨).

```
// Pull credentials from the default provider chain
$provider = Aws\Credentials\CredentialProvider::defaultProvider();
$credentials = call_user_func($provider)->wait();

// Create a signer with the service's signing name and Region
$signer = new Aws\Signature\SignatureV4('es', 'us-west-2');

// Sign your request
$signedRequest = $signer->signRequest($psr7Request, $credentials);
```

AWS SDK for PHP 버전 3을 사용한 AWS Identity and Access Management 예제

AWS Identity and Access Management(IAM)는 Amazon Web Services 고객이 AWS의 사용자와 각 사용자 권한을 관리할 수 있도록 하는 웹 서비스입니다. 이 서비스는 클라우드 내에 AWS 제품을 사용하는 여러 사용자 또는 시스템이 있는 조직을 대상으로 합니다. IAM을 사용하면 사용자, 액세스 키와 같은 보안 자격 증명, 사용자가 액세스할 수 있는 AWS 리소스를 제어하는 권한을 중앙 관리할 수 있습니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용한 IAM 액세스 키 관리

- AWS SDK for PHP 버전 3으로 IAM 사용자 관리
- AWS SDK for PHP 버전 3에서 IAM 계정 별칭 사용
- AWS SDK for PHP 버전 3에서 IAM 정책 사용
- AWS SDK for PHP 버전 3을 사용한 IAM 서버 인증서 작업

AWS SDK for PHP 버전 3을 사용한 IAM 액세스 키 관리

사용자가 AWS를 프로그래밍 방식으로 호출하려면 고유의 액세스 키가 필요합니다. 이 요구를 충족하기 위해 IAM 사용자에 대한 액세스 키(액세스 키 ID 및 보안 액세스 키)를 생성, 수정, 확인 또는 교체할 수 있습니다. 기본적으로 액세스 키를 생성할 때 키의 상태는 활성입니다. 따라서 사용자는 액세스 키를 API 호출에 사용할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateAccessKey를 사용하여 보안 액세스 키 및 해당 액세스 키 ID를 생성합니다.
- ListAccessKeys를 사용하여 IAM 사용자와 연결된 액세스 키 ID에 대한 정보를 반환합니다.
- GetAccessKeyLastUsed를 사용하여 액세스 키가 마지막을 사용되었을 때에 대한 정보를 검색합니다.
- UpdateAccessKey를 사용하여 액세스 키의 상태를 활성에서 비활성으로 변경하거나 반대로 변경합니다.
- DeleteAccessKey를 사용하여 IAM 사용자와 연결된 액세스 키 쌍을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

액세스 키 생성

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

액세스 키 나열

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

액세스 키의 마지막 사용에 대한 정보 가져오기

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

액세스 키 업데이트

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

액세스 키 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->deleteAccessKey([
        'AccessKeyId' => 'ACCESS_KEY_ID', // REQUIRED
        'UserName' => 'IAM_USER_NAME',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3으로 IAM 사용자 관리

IAM 사용자는 AWS에서 생성하는 개체로서 AWS와 상호 작용하기 위해 그 개체를 사용하는 사람 또는 서비스를 대표합니다. AWS에서 사용자는 이름과 보안 인증으로 구성됩니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateUser를 사용하여 새 IAM 사용자를 생성합니다.
- ListUsers를 사용하여 IAM 사용자를 나열합니다.
- UpdateUser를 사용하여 IAM 사용자를 업데이트합니다.
- GetUser를 사용하여 IAM 사용자에 대한 정보를 검색합니다.
- DeleteUser를 사용하여 IAM 사용자를 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

IAM 사용자를 생성합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->createUser(array(
        // UserName is required
        'UserName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM 사용자 나열

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->listUsers();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM 사용자 생성

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->updateUser([
        // UserName is required
        'UserName' => 'string1',
        'NewUserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM 사용자 정보 조회

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->getUser([
        'UserName' => 'string',
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

IAM 사용자 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
```

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

샘플 코드

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

try {
    $result = $client->deleteUser([
        // UserName is required
        'UserName' => 'string'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3에서 IAM 계정 별칭 사용

AWS 계정 ID 대신 회사 이름이나 기타 친숙한 식별자를 로그인 페이지의 URL에 포함하려는 경우 AWS 계정 ID의 별칭을 만들 수 있습니다. AWS 계정 계정 별칭을 생성할 경우 명칭을 적용하기 위해 로그인 페이지 URL이 변경됩니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateAccountAlias를 사용하여 별칭을 생성합니다.
- ListAccountAliases를 사용하여 AWS 계정과 연결된 별칭을 나열합니다.
- DeleteAccountAlias를 사용하여 별칭을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

별칭 만들기

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

계정 별칭 조회

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

별칭 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->deleteAccountAlias([
        // AccountAlias is required
        'AccountAlias' => 'string',
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3에서 IAM 정책 사용

정책을 생성하여 사용자에게 권한을 부여합니다. 정책은 사용자가 수행할 수 있는 작업과 작업이 적용되는 리소스를 나열하는 문서입니다. 기본적으로 명시적으로 허용되지 않은 작업 또는 리소스는 모두 거부됩니다. 정책을 생성하여 사용자, 사용자 그룹, 사용자가 맡는 역할, 리소스에 연결할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreatePolicy를 사용하여 관리형 정책을 생성합니다.
- AttachRolePolicy를 사용하여 정책을 역할에 연결합니다.
- AttachUserPolicy를 사용하여 정책을 사용자에 연결합니다.
- AttachGroupPolicy를 사용하여 정책을 그룹에 연결합니다.
- DetachRolePolicy를 사용하여 역할 정책을 제거합니다.
- DetachUserPolicy를 사용하여 사용자 정책을 제거합니다.
- DetachGroupPolicy를 사용하여 그룹 정책을 제거합니다.
- DeletePolicy를 사용하여 관리형 정책을 삭제합니다.
- DeleteRolePolicy를 사용하여 역할 정책을 삭제합니다.
- DeleteUserPolicy를 사용하여 사용자 정책을 삭제합니다.
- DeleteGroupPolicy를 사용하여 그룹 정책을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

정책 생성

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

역할에 정책 연결

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

사용자에게 정책 연결

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

정책을 그룹에 연결

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

사용자 정책 분리

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

그룹 정책 분리

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

정책 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

역할 정책 제거

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

사용자 정책 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
```

```
}
```

그룹 정책 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

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

try {
    $result = $client->deleteGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용한 IAM 서버 인증서 작업

AWS에서 웹 사이트나 애플리케이션에 대한 HTTPS 연결을 활성화하려면 SSL/TLS 서버 인증서가 필요합니다. 외부 공급자에게서 얻은 인증서를 AWS의 웹 사이트 또는 애플리케이션에서 사용하려면 해당 인증서를 IAM에 업로드하거나 AWS Certificate Manager로 가져와야 합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListServerCertificates를 사용하여 IAM에 저장된 인증서를 나열합니다.
- GetServerCertificate를 사용하여 인증서에 대한 정보를 검색합니다.
- UpdateServerCertificate를 사용하여 인증서를 업데이트합니다.
- DeleteServerCertificate를 사용하여 인증서를 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

서버 인증서 나열

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

서버 인증서 검색

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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());
}
```

서버 인증서 업데이트

가져옵니다.

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

use Aws\Exception\AwsException;
```

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

샘플 코드

```
$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());
}
```

서버 인증서 삭제

가져옵니다.

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Iam\IamClient;
```

샘플 코드

```
$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 SDK for PHP 버전 3을 사용한 AWS Key Management Service 예제

AWS Key Management Service(AWS KMS)는 데이터 암호화에 사용하는 암호화 키를 쉽게 생성하고 제어할 수 있게 해주는 관리형 서비스입니다. AWS KMS에 대한 자세한 내용은 Amazon KMS 설명서를 참조하세요. 보안 PHP 애플리케이션을 작성하든 또는 다른 AWS 서비스에 데이터를 전송하든 상관없이 AWS KMS를 통해 마스터 키를 사용해 암호화된 데이터에 대한 액세스할 수 있는 사용자를 관리할 수 있습니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 키 사용
- AWS KMS 버전 3을 사용한 AWS SDK for PHP 데이터 키 암호화 및 복호화
- AWS SDK for PHP 버전 3을 사용한 AWS KMS 주요 정책 관련 작업
- AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 보조금 관련 작업
- AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 별칭 작업

AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 키 사용

AWS Key Management Service(AWS KMS)의 기본 리소스는 AWS KMS keys입니다. KMS 키를 사용하여 데이터를 암호화할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateKey를 사용하여 고객 KMS 키를 생성합니다.

- GenerateDataKey를 사용하여 데이터 키를 생성합니다.
- DescribeKey를 사용하여 KMS 키를 확인합니다.
- ListKeys를 사용하여 KMS 키의 키 ID와 키 ARN을 확인합니다.
- EnableKey를 사용하여 KMS 키를 활성화합니다.
- DisableKey를 사용하여 KMS 키를 비활성화합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하세요.

KMS 키 생성

KMS 키를 생성하려면 CreateKey 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

데이터 키 생성

데이터 암호화 키를 생성하려면 GenerateDataKey 작업을 사용합니다. 이 작업은 생성되는 일반 텍스트와 암호화된 데이터 키 사본을 반환합니다. 데이터 키를 생성할 때 사용할 AWS KMS key를 지정합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$keySpec = 'AES_256';

try {
    $result = $KmsClient->generateDataKey([
        'KeyId' => $keyId,
        'KeySpec' => $keySpec,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

KMS 키 보기

KMS 키의 Amazon 리소스 이름(ARN) 및 키 상태를 비롯해 KMS 키에 대한 자세한 정보를 가져오려면 DescribeKey 작업을 사용합니다.

DescribeKey는 별칭을 가져오지 않습니다. 별칭을 가져오려면 ListAliases 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->describeKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
```

```
}
```

KMS 키의 키 ID와 키 ARN 확인

KMS의 ID와 ARN을 확인하려면 ListAliases 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$limit = 10;

try {
    $result = $KmsClient->listKeys([
        'Limit' => $limit,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

KMS 키가 활성화됨

비활성화된 KMS 키를 활성화하려면 EnableKey 작업을 이용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

비활성화된 KMS 키

KMS 키를 비활성화하려면 DisableKey 작업을 이용합니다. KMS 키를 비활성화하면 삭제할 수 없습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';

try {
    $result = $KmsClient->disableKey([
        'KeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS KMS 버전 3을 사용한 AWS SDK for PHP 데이터 키 암호화 및 복호화

데이터 키는 많은 양의 데이터 및 기타 데이터 암호화 키를 포함하여 데이터를 암호화하는 데 사용할 수 있는 암호화 키입니다.

데이터 키의 생성, 암호화 및 복호화를 위해 AWS Key Management Service의 (AWS KMS)AWS KMS key를 사용할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- 암호화를 사용하여 데이터 키를 암호화합니다.
- 암호화 해제를 사용하여 데이터 키를 암호화 해제합니다.
- ReEncrypt를 사용하여 새로운 KMS 키로 데이터 키를 다시 암호화합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하세요.

Encrypt

암호화 작업은 데이터 키를 암호화하도록 설계되었지만 자주 사용되지 않습니다. GenerateDataKey 및 GenerateDataKeyWithoutPlaintext 작업은 암호화된 데이터 키를 반환합니다. 암호화된 데이터를 새로운 AWS 리전으로 이동하고 새 리전의 KMS 키를 사용하여 데이터 키를 암호화하려는 경우 `Encypt` 메서드를 사용할 수 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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

데이터 키를 해독하려면 Decrypt 작업을 사용합니다.

지정하는 `ciphertextBlob`는 GenerateDataKey, GenerateDataKeyWithoutPlaintext 또는 암호화 응답의 `CiphertextBlob` 필드 값이어야 합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

재암호화

암호화한 데이터 키를 해독한 후 다른 KMS 키에서 즉시 데이터 키를 재암호화하려면 ReEncrypt 작업을 사용합니다. 이러한 작업은 모두 AWS KMS 내부의 서버 측에서 수행되므로 AWS KMS 외부에 일반 텍스트를 노출해서는 안 됩니다.

지정하는 ciphertextBlob는 GenerateDataKey, GenerateDataKeyWithoutPlaintext 또는 암호화 응답의 CiphertextBlob 필드 값이어야 합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$ciphertextBlob = 'Place your cipher text blob here';

try {
    $result = $KmsClient->reEncrypt([
        'CiphertextBlob' => $ciphertextBlob,
        'DestinationKeyId' => $keyId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 AWS KMS 주요 정책 관련 작업

AWS KMS key를 만들 때 KMS 키를 사용하고 관리할 수 있는 담당자를 결정합니다. 이러한 권한은 키 정책이라는 문서에 포함됩니다. 키 정책을 사용하여 고객 관리형 KMS 키에 대한 권한을 언제든 추가, 제거 또는 변경할 수 있지만 AWS 관리형 KMS 키에 대한 키 정책은 편집할 수 없습니다. 자세한 내용은 AWS KMS에 대한 인증 및 액세스 제어를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListKeyPolicies를 사용하여 키 정책 이름의 목록을 표시합니다.
- GetKeyPolicy를 사용하여 키 정책을 확인합니다.
- PutKeyPolicy를 사용하여 키 정책을 설정합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하세요.

모든 키 정책의 목록 표시

KMS 키의 키 정책 이름을 확인하려면 `ListKeyPolicies` 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

키 정책 검색

KMS 키의 키 정책을 확인하려면 `GetKeyPolicy` 작업을 사용합니다.

`GetKeyPolicy`는 정책 이름을 요구합니다. KMS 키를 생성할 때 키 정책을 생성하지 않은 경우 유효한 정책 이름은 기본 이름뿐입니다. AWS Key Management Service개발자 안내서에서 기본 키 정책에 대해 자세히 알아보세요.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

키 정책 설정

KMS 키에 대한 키 정책을 설정하거나 변경하려면 PutKeyPolicy 작업을 사용합니다.

PutKeyPolicy는 정책 이름을 요구합니다. KMS 키를 생성할 때 키 정책을 생성하지 않은 경우 유효한 정책 이름은 기본 이름뿐입니다. AWS Key Management Service개발자 안내서에서 기본 키 정책에 대해 자세히 알아보세요.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$policyName = "default";

try {
    $result = $KmsClient->putKeyPolicy([
        'KeyId' => $keyId,
        'PolicyName' => $policyName,
        'Policy' => '{
            "Version":"2012-10-17",
```

```
            "Id": "custom-policy-2016-12-07",
            "Statement": [
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/root" },
                "Action": [ "kms:*" ],
                "Resource": "*" },
                { "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal":
                   { "AWS": "arn:aws:iam::111122223333:user/ExampleUser" },
                "Action": [
                    "kms:Encrypt*",
                    "kms:GenerateDataKey*",
                    "kms:Decrypt*",
                    "kms:DescribeKey*",
                    "kms:ReEncrypt*"
                ],
                "Resource": "*" }
            ]
        } '
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 보조금 관련 작업

권한 부여는 권한을 제공하기 위한 또 하나의 메커니즘입니다. 이는 키 정책의 대안입니다. 권한 부여를 이용해 AWS 보안 주체가 AWS Key Management Service(AWS KMS) 고객 관리형 AWS KMS keys를 사용하도록 허용하는 장기 액세스 권한을 부여할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 가이드에서 AWS KMS 권한 부여를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateGrant를 사용하여 KMS 키에 대한 권한 부여를 생성합니다.
- ListGrants를 사용하여 KMS 키에 대한 권한 부여를 봅니다.

- RetireGrant를 사용하여 KMS 키에 대한 권한 부여를 사용 중지시킵니다.
- RevokeGrant를 사용하여 KMS 키에 대한 권한 부여를 취소합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하세요.

권한 부여 생성

AWS KMS key에 대한 권한 부여를 생성하려면 CreateGrant 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

권한 부여 보기

AWS KMS key에 대한 권한 부여를 자세히 알아보려면 ListGrants 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
```

```
}
```

권한 부여 사용 중지

AWS KMS key에 대한 권한 부여를 사용 중지하려면 RetireGrant 작업을 사용합니다. 권한 부여의 사용을 완료한 후에는 만료시킵니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

권한 부여 취소

AWS KMS key에 대한 권한 부여를 취소하려면 RevokeGrant 작업을 사용합니다. 권한 부여를 취소하여 종속된 작업을 명시적으로 거부할 수 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$grantId = "grant1";

try {
    $result = $KmsClient->revokeGrant([
        'KeyId' => $keyId,
        'GrantId' => $grantId,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS KMS API 및 AWS SDK for PHP 버전 3을 사용한 별칭 작업

AWS Key Management Service(AWS KMS)는 AWS KMS key 호출된 별칭의 선택적 표시 이름을 제공합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateAlias를 사용하여 별칭을 생성합니다.
- ListAliases를 사용하여 별치 보기
- UpdateAlias를 사용하여 별칭을 업데이트합니다.
- DeleteAlias를 사용하여 별칭을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하세요.

별칭 만들기

KMS 키에 대한 별칭을 생성하려면 CreateAlias 작업을 사용합니다. 별칭은 계정 및 AWS 리전 내에서 고유해야 합니다. 이미 별칭이 있는 KMS 키에 별칭을 생성할 경우, `CreateAlias`가 동일한 KMS 키에 대해 다른 별칭을 생성합니다. 새 별칭이 기존 별칭을 대체하지는 않습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

별칭 보기

호출자의 AWS 계정 및 AWS 리전에서 별칭을 나열하려면 ListAliases 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

별칭 업데이트

기존 별칭을 다른 KMS 키와 연결하려면 UpdateAlias 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

별칭 삭제

별칭을 삭제하려면 DeleteAlias 작업을 사용합니다. 별칭을 삭제해도 기본 KMS 키에 영향을 미치지 않습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$aliasName = "alias/projectKey1";

try {
    $result = $KmsClient->deleteAlias([
        'AliasName' => $aliasName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon Kinesis 예제

Amazon Kinesis는 데이터를 실시간으로 수집, 처리, 분석하는 AWS 서비스입니다. Amazon Kinesis Data Streams로 데이터 스트림을 구성하거나 Amazon Data Firehose를 사용하여 데이터를 Amazon S3, OpenSearch Service, Amazon Redshift 또는 Splunk로 전송할 수 있습니다.

Kinesis에 대한 자세한 내용은 Amazon Kinesis 문서를 참조하세요.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 스트림 생성
- Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 샤드 관리
- Firehose API 및 AWS SDK for PHP 버전 3을 사용하여 전송 스트림 생성

Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 스트림 생성

Amazon Kinesis Data Streams를 사용하면 실시간 데이터를 전송할 수 있습니다. Kinesis Data Streams를 사용하여 데이터를 추가할 때마다 구성한 대상으로 데이터를 전송하는 데이터 생산자를 생성합니다.

자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서의 스트림 생성 및 관리를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateAlias를 사용하여 데이터 스트림을 생성합니다.
- DescribeStream을 사용하여 단일 데이터 스트림에 대한 세부 정보를 확인합니다.
- ListStreams를 사용하여 기존 데이터 스트림의 목록을 표시합니다.
- PutRecord를 사용하여 기존 데이터 스트림으로 데이터를 전송합니다.
- DeleteStream을 사용하여 데이터 스트림을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon Kinesis 개발자 안내서 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서를 참조하세요.

Kinesis 데이터 스트림을 사용하여 데이터 스트림 생성

다음 코드 예제를 사용하여 Kinesis에서 처리할 정보를 전송할 수 있는 Kinesis 데이터 스트림을 설정합니다. Amazon Kinesis 개발자 안내서에서 데이터 스트림 생성 및 업데이트에 대해 자세히 알아보세요.

Kinesis 데이터 스트림을 생성하려면, CreateStream 작업을 사용하세요.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

데이터 스트림 검색

다음 코드 예제를 사용하여 기존 데이터 스트림에 대한 세부 정보를 확인할 수 있습니다. 기본적으로 지정한 Kinesis 데이터 스트림에 연결된 처음 10개 샤드에 대한 정보가 반환됩니다. Kinesis 데이터 스트림에 쓰기 전에 응답의 `StreamStatus`를 확인하세요.

지정한 Kinesis 데이터 스트림에 대한 세부 정보를 확인하려면 DescribeStream 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

Kinesis에 연결된 기존 데이터 스트림의 목록 표시

선택한 AWS 리전에서 해당 AWS 계정 계정의 처음 10개 데이터 스트림의 목록을 표시합니다. 반환된 `HasMoreStreams를 사용하여 계정에 더 많은 스트림이 연결되었는지를 확인합니다.

Kinesis 데이터 스트림의 목록을 표시하려면 ListStreams 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

기존 데이터 스트림에 데이터 전송

데이터 스트림을 생성한 후 다음 예제를 사용하여 데이터를 전송합니다. 데이터를 전송하기 전에 DescribeStream을 사용하여 데이터의 StreamStatus가 활성인지 여부를 확인합니다.

Kinesis 데이터 스트림에 단일 데이터 레코드를 기록하려면 PutRecord 작업을 사용합니다. Kinesis 데이터 스트림에 최대 500개의 레코드를 기록하려면 PutRecords 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

데이터 스트림 삭제

이 예제는 데이터 스트림을 삭제하는 방법을 보여줍니다. 데이터를 삭제하면 데이터 스트림으로 전송한 데이터도 삭제됩니다. 활성 Kinesis 데이터 스트림은 스트림 삭제가 완료될 때까지 DELETING 상태로 전환됩니다. DELETING 상태에서는 스트림에서 데이터 처리를 계속합니다.

Kinesis 데이터 스트림을 삭제하려면, DeleteStream 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $kinesisClient->deleteStream([
        'StreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 샤드 관리

Amazon Kinesis Data Streams를 사용하면 엔드포인트로 실시간 데이터를 엔드포인트로 전송할 수 있습니다. 데이터 흐름 속도는 스트림의 샤드 수에 따라 다릅니다.

한 샤드에 초당 1,000개의 레코드를 쓸 수 있습니다. 또한 각 샤드의 업로드 제한 속도는 초당 1MiB입니다. 사용은 샤드 단위로 계산되고 요금이 청구되므로, 다음 예제를 사용하여 스트림의 데이터 용량과 비용을 관리하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListShards를 사용하여 스트림의 샤드 목록을 표시합니다.

- UpdateShardCount를 사용하여 스트림의 샤드 수를 추가하거나 줄입니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon Kinesis Data Streams 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서를 참조하세요.

데이터 스트림 샤드 목록 표시

특정 스트림의 최대 100개 샤드에 대한 세부 정보를 표시합니다.

Kinesis Data Streams 데이터 스트림의 샤드 목록을 표시하려면 ListShards 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

데이터 스트림 샤드 추가

데이터 스트림 샤드가 더 필요할 경우 현재 샤드 수를 늘릴 수 있습니다. 증가 시 샤드 수를 두 배로 늘리는 것이 좋습니다. 이렇게 하면 현재 사용할 수 있는 각 샤드의 복사본이 생성되어 용량이 늘어납니다. 샤드 수 두 배 증가는 24시간 동안 두 번만 할 수 있습니다.

Kinesis Data Streams 사용량에 대한 청구서는 샤드당 계산되므로 수요가 감소하면 샤드 수를 절반으로 줄이는 것이 좋습니다. 샤드를 제거할 때 현재 샤드 수의 절반까지만 샤드의 양을 줄일 수 있습니다.

Kinesis Data Streams 데이터 스트림의 샤드 수를 업데이트하려면 UpdateShardCount 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$kinesisClient = new Aws\Kinesis\KinesisClient([
    'profile' => 'default',
    'version' => '2013-12-02',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$totalshards = 4;

try {
    $result = $kinesisClient->UpdateShardCount([
        'ScalingType' => 'UNIFORM_SCALING',
        'StreamName' => $name,
        'TargetShardCount' => $totalshards
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Firehose API 및 AWS SDK for PHP 버전 3을 사용하여 전송 스트림 생성

Amazon Data Firehose를 사용하면 Amazon Kinesis Data Streams, Amazon S3, Amazon OpenSearch Service (OpenSearch Service), Amazon Redshift 등의 다른 AWS 서비스 또는 Splunk로 실시간 데이터를 전송할 수 있습니다. 전송 스트림으로 데이터 생산자를 생성하여 데이터를 추가할 때마다 구성한 대상으로 데이터를 전송합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateDeliveryStream을 사용하여 전송 스트림을 생성합니다.
- DescribeDeliveryStream을 사용하여 단일 전송 스트림에 대한 세부 정보를 확인합니다.
- ListDeliveryStreams를 사용하여 전송 스트림 목록을 표시합니다.
- PutRecord를 사용하여 전송 스트림으로 데이터를 전송합니다.
- DeleteDeliveryStream을 사용하여 전송 스트림을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Amazon Data Firehose 사용에 대한 자세한 내용은 Amazon Kinesis Data Firehose 개발자 설명서를 참조세요.

Kinesis 데이터 스트림을 사용하여 전송 스트림 생성

데이터를 기존 Kinesis 데이터 스트림에 넣는 전송 스트림을 설정하려면 CreateDeliveryStream 작업을 사용합니다.

이를 통해 개발자는 기존 Kinesis 서비스를 Firehouse로 마이그레이션할 수 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$stream_type = "KinesisStreamAsSource";
$kinesis_stream = "arn:aws:kinesis:us-east-2:0123456789:stream/my_stream_name";
$role = "arn:aws:iam::0123456789:policy/Role";

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'KinesisStreamSourceConfiguration' => [
            'KinesisStreamARN' => $kinesis_stream,
            'RoleARN' => $role,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon S3 버킷을 사용하여 전송 스트림 생성

데이터를 기존 Amazon S3 버킷에 넣는 전송 스트림을 설정하려면 CreateDeliveryStream 작업을 사용합니다.

대상 파라미터에서 설명한 대로 대상 파라미터를 제공합니다. 그런 다음 Kinesis Data Firehose에 Amazon S3 대상 액세스 권한 부여에 설명된 대로 Firehose가 Amazon S3 버킷에 액세스할 수 있도록 허용하세요.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_S3_stream_name";
$stream_type = "DirectPut";
$s3bucket = 'arn:aws:s3:::bucket_name';
$s3Role = 'arn:aws:iam::0123456789:policy/Role';

try {
    $result = $firehoseClient->createDeliveryStream([
        'DeliveryStreamName' => $name,
        'DeliveryStreamType' => $stream_type,
        'S3DestinationConfiguration' => [
            'BucketARN' => $s3bucket,
            'CloudWatchLoggingOptions' => [
                'Enabled' => false,
            ],
            'RoleARN' => $s3Role
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

OpenSearch 서비스를 사용하여 전송 스트림 생성

데이터를 OpenSearch Service 클러스터로 전송하는 Firehose 전송 스트림을 설정하려면 CreateDeliveryStream 작업을 사용합니다.

대상 파라미터에서 설명한 대로 대상 파라미터를 제공합니다. Kinesis Data Firehose에 Amazon ES 대상 액세스 권한 부여에 설명된 대로 Firehose가 OpenSearch Service 클러스터에 액세스할 수 있도록 허용하세요.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

전송 스트림 검색

기존 Firehose 전송 시스템에 대한 세부 정보를 확인하려면 DescribeDeliveryStream 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
```

```
try {
    $result = $firehoseClient->describeDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Kinesis Data Streams에 연결된 기존 전송 스트림 목록 표시

데이터를 Kinesis Data Streams로 전송하는 기존의 모든 Firehose 전송 스트림을 나열하려면 ListDeliveryStreams 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

다른 AWS 서비스로 데이터를 전송하는 기존 전송 스트림의 목록 표시

Amazon S3, OpenSearch Service 또는 Amazon Redshift 또는 Splunk로 데이터를 전송하는 기존의 모든 Firehose 전송 스트림을 나열하려면 ListDeliveryStreams 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

기존 Firehose 전송 스트림으로 데이터 전송

Firehose 전송 스트림을 통해 데이터를 지정된 대상으로 전송하려면 Firehose 전송 스트림을 생성한 후 PutRecord 작업을 사용합니다.

Firehose 전송 스트림으로 데이터를 전송하기 전에 `DescribeDeliveryStream`을 사용하여 전송 스트림이 활성 상태인지 확인합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";
$content = '{"ticker_symbol":"QXZ", "sector":"HEALTHCARE", "change":-0.05,
 "price":84.51}';

try {
    $result = $firehoseClient->putRecord([
        'DeliveryStreamName' => $name,
        'Record' => [
            'Data' => $content,
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Firehose 전송 스트림 삭제

Firehose 전송 스트림을 삭제하려면 DeleteDeliveryStreams 작업을 사용합니다. 이 작업은 전송 스트림으로 전송한 데이터도 삭제합니다.

가져옵니다.

```
```

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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 SDK for PHP 버전 3을 사용한 AWS Elemental MediaConvert 예제

AWS Elemental MediaConvert는 브로드캐스트급 기능을 갖춘 파일 기반의 비디오 트랜스코딩 서비스입니다. 이 서비스를 사용하여 인터넷에서 브로드캐스트 및 비디오 온디맨드(VOD) 전달용 자산을 생성할 수 있습니다. 자세한 내용은 AWS Elemental MediaConvert 사용 설명서를 참조하세요.

AWS Elemental MediaConvert용 PHP API는 *`AWS.MediaConvert`* 클라이언트 클래스를 통해 노출됩니다. 자세한 내용은 API 참조의 `Class: AWS.MediaConvert`를 참조하세요.

AWS Elemental MediaConvert에서 트랜스코딩 작업 생성 및 관리

이 예제에서는 AWS SDK for PHP 버전 3을 사용하여 AWS Elemental MediaConvert를 호출한 후 트랜스코딩 작업을 생성합니다. 시작하기 전에 입력 스토리지로 프로비저닝한 Amazon S3 버킷에 입력 비디오를 업로드해야 합니다. 지원되는 입력 비디오 코덱 및 컨테이너 목록은 AWS Elemental MediaConvert 사용 설명서의 지원되는 입력 코덱 및 컨테이너를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- AWS Elemental MediaConvert에서 트랜스코딩 작업을 생성합니다. CreateJob.
- AWS Elemental MediaConvert 대기열에서 트랜스코딩 작업을 취소합니다. CancelJob
- 완료된 트랜스코딩 작업에 대한 JSON을 검색합니다. GetJob
- 최근에 생성된 최대 20개 작업에 대한 JSON 배열을 검색합니다. ListJobs

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

MediaConvert 클라이언트에 액세스하려면 출력 파일이 저장된 Amazon S3 버킷 및 입력 파일에 대한 AWS Elemental MediaConvert 액세스 권한을 부여하는 IAM 역할을 만듭니다. 자세한 내용은 AWS Elemental MediaConvert 사용자 가이드의 IAM 권한 설정하기를 참조하세요.

클라이언트 만들기

코드에 사용할 리전으로 MediaConvert 클라이언트를 생성하여 AWS SDK for PHP를 구성합니다. 예를 들면, 리전이 us-west-2로 설정되어 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

샘플 코드

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

단순 트랜스코딩 작업 정의

트랜스코드 작업 파라미터를 정의하는 JSON을 생성합니다.

이러한 파라미터는 세부적으로 정의됩니다. AWS Elemental MediaConvert 콘솔을 사용하면 콘솔에서 작업 설정을 선택한 다음 작업 섹션 하단에서 작업 JSON 표시를 선택하여 JSON 작업 파라미터를 생성할 수 있습니다. 이 예제는 단순 작업용 JSON을 보여줍니다.

샘플 코드

```
$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"
```

```
    ]
];
```

작업 만들기

작업 파라미터 JSON을 생성한 후에는 AWS.MediaConvert service object를 호출하고 파라미터를 전달하여 createJob 메서드를 호출합니다. 생성된 작업의 ID는 응답 데이터로 반환됩니다.

샘플 코드

```
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";
}
```

작업 검색

createjob 호출 시 반환되었던 JobID를 사용하여 JSON 형식의 최근 작업에 대한 자세한 설명을 가져올 수 있습니다.

샘플 코드

```
$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";
}
```

작업 취소

createjob 호출 시 반환되었던 JobID를 사용하여 대기열에 있는 동안 작업을 취소할 수 있습니다. 이미 트랜스코딩이 시작된 작업은 취소할 수 없습니다.

샘플 코드

```
$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";
}
```

최근 트랜스코딩 작업 나열

목록을 오름차순 또는 내림차순으로 정렬할지 여부, 확인할 작업 대기열의 ARN 및 포함할 작업 상태를 지정하는 값을 포함하여 JSON 파라미터를 생성합니다. 그러면 최대 20개 작업이 반환됩니다. 그 다음 최근 작업 20개를 가져오려면 결과로 반환되는 nextToken 문자열을 사용합니다.

샘플 코드

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
```

```
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->listJobs([
        'MaxResults' => 20,
        'Order' => 'ASCENDING',
        'Queue' => 'QUEUE_ARN',
        'Status' => 'SUBMITTED',
        // 'NextToken' => '<string>', //OPTIONAL To retrieve the twenty next most
 recent jobs
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3 예제

Amazon Simple Storage Service (Amazon S3)는 안전하고 내구성과 확장성이 뛰어난 클라우드 스토리지를 제공합니다. Amazon S3는 웹 어디에서나 원하는 양의 데이터를 저장하고 검색할 수 있는 간단한 웹 서비스 인터페이스를 갖춘 사용하기 쉬운 객체 스토리지입니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3에서 Amazon S3 버킷 생성 및 사용
- AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷 액세스 권한 관리
- AWS SDK for PHP 버전 3에서 Amazon S3 버킷 구성
- AWS SDK for PHP 버전 3에서 Amazon S3 멀티파트 업로드 사용
- AWS SDK for PHP 버전 3을 사용한 Amazon S3의 사전 서명된 URL
- AWS SDK for PHP 버전 3을 사용한 Amazon S3에서 사전 서명된 게시물
- AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷을 정적 웹 호스트로 사용

- AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷 정책 작업
- S3 액세스 포인트 ARN AWS SDK for PHP 버전 3사용
- AWS SDK for PHP 버전 3에서 Amazon S3 다중 리전 액세스 포인트 사용

AWS SDK for PHP 버전 3에서 Amazon S3 버킷 생성 및 사용

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListBuckets을 사용하여 요청의 인증된 발신자가 소유한 버킷 목록을 반환합니다.
- CreateBucket을 사용하여 새 버킷을 생성합니다.
- PutObject를 사용하여 버킷에 객체를 추가합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

가져옵니다.

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

use Aws\S3\S3Client;
```

버킷 나열

다음 코드를 사용하여 PHP 파일을 생성합니다. 먼저 AWS 리전과 버전을 지정하는 AWS.S3 클라이언트 서비스를 생성합니다. 그런 다음 요청 발신자가 소유한 모든 Amazon S3 버킷을 버킷 구조 배열로 반환하는 `listBuckets` 메서드를 호출합니다.

샘플 코드

```
$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";
}
```

버킷 생성

다음 코드를 사용하여 PHP 파일을 생성합니다. 먼저 AWS 리전과 버전을 지정하는 AWS.S3 클라이언트 서비스를 생성합니다. 그런 다음 array를 파라미터로 사용하여 `createBucket` 메서드를 호출합니다. 유일하게 필요한 필드는 생성하려는 버킷 이름에 대한 문자열 값을 포함하는 'Bucket' 키입니다. 하지만 'CreateBucketConfiguration' 필드를 사용하여 AWS 리전을 지정할 수 있습니다. 성공할 경우 이 메서드는 버킷의 '위치'를 반환합니다.

샘플 코드

```
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();
```

버킷에 객체 넣기

새 버킷에 파일을 추가하려면 다음 코드를 사용하여 PHP 파일을 생성합니다.

명령줄에서 이 파일을 실행하고 파일을 업로드할 버킷의 이름을 문자열로 전달하고 그 뒤에 업로드하려는 파일의 전체 파일 경로를 입력합니다.

샘플 코드

```
$USAGE = "\n" .
    "To run this example, supply the name of an S3 bucket and a file to\n" .
    "upload to it.\n" .
    "\n" .
    "Ex: php PutObject.php <bucketname> <filename>\n";

if (count($argv) <= 2) {
    echo $USAGE;
    exit();
}

$bucket = $argv[1];
$file_Path = $argv[2];
$key = basename($argv[2]);

try {
    //Create a S3Client
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-west-2',
        'version' => '2006-03-01'
    ]);
    $result = $s3Client->putObject([
        'Bucket' => $bucket,
        'Key' => $key,
        'SourceFile' => $file_Path,
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷 액세스 권한 관리

ACL(액세스 제어 목록)은 리소스 기반 액세스 정책 옵션 중 하나로, 해당 옵션을 사용해 버킷과 객체에 대한 액세스를 관리할 수 있습니다. ACL로 다른 AWS 계정에 기본적인 읽기/쓰기 권한을 부여할 수 있습니다. 자세한 내용은 ACL을 사용한 액세스 관리를 참조하세요.

다음 예에서는 작업 방법을 보여줍니다.

- GetBucketAcl을 사용하여 버킷에 대한 액세스 제어 정책을 가져옵니다.
- PutBucketAcl을 사용하여 ACL로 버킷에 대한 권한을 설정합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

액세스 제어 목록 정책 가져오기 및 설정

가져옵니다.

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

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

샘플 코드

```

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

// Gets the access control policy for a bucket
$bucket = 'amzn-s3-demo-bucket';
try {
```

```
    $resp = $s3Client->getBucketAcl([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving bucket ACL as follows: \n";
    var_dump($resp);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Sets the permissions on a bucket using access control lists (ACL).
$params = [
    'ACL' => 'public-read',
    'AccessControlPolicy' => [
        // Information can be retrieved from `getBucketAcl` response
        'Grants' => [
            [
                'Grantee' => [
                    'DisplayName' => '<string>',
                    'EmailAddress' => '<string>',
                    'ID' => '<string>',
                    'Type' => 'CanonicalUser',
                    'URI' => '<string>',
                ],
                'Permission' => 'FULL_CONTROL',
            ],
            // ...
        ],
        'Owner' => [
            'DisplayName' => '<string>',
            'ID' => '<string>',
        ],
    ],
    'Bucket' => $bucket,
];

try {
    $resp = $s3Client->putBucketAcl($params);
    echo "Succeed in setting bucket ACL.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
```

```
}
```

AWS SDK for PHP 버전 3에서 Amazon S3 버킷 구성

교차 오리진 리소스 공유(CORS) 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. Amazon S3에서 CORS 지원을 통해 Amazon S3으로 다양한 기능의 클라이언트 측 웹 애플리케이션을 구축하고, Amazon S3 리소스에 대한 Cross-Origin 액세스를 선택적으로 허용할 수 있습니다.

Amazon S3 버킷에서 CORS 구성을 사용하는 방법에 대한 자세한 내용은 Cross-Origin Resource Sharing (CORS)을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- GetBucketCors를 사용하여 버킷에 대한 CORS 구성을 가져옵니다.
- PutBucketCors를 사용하여 버킷에 대한 CORS 구성을 설정합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

CORS 구성 가져오기

다음 코드를 사용하여 PHP 파일을 생성합니다. 먼저 AWS.S3 클라이언트 서비스를 생성한 다음 `getBucketCors` 메서드를 호출하고 CORS 구성을 원하는 버킷을 지정합니다.

필요한 유일한 파라미터는 선택된 버킷의 이름입니다. 버킷에 현재 CORS 구성이 있는 경우 Amazon S3에서 해당 구성을 CORSRules 객체로 반환합니다.

가져옵니다.

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

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

샘플 코드

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

try {
    $result = $client->getBucketCors([
        'Bucket' => $bucketName, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

CORS 구성 설정

다음 코드를 사용하여 PHP 파일을 생성합니다. 먼저 AWS.S3 클라이언트 서비스를 생성합니다. 그런 다음 putBucketCors 메서드를 호출하고 CORS 구성을 설정할 버킷을 지정하고 CORSConfiguration을 CORSRules JSON 객체로 지정합니다.

가져옵니다.

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

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

샘플 코드

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

```
try {
    $result = $client->putBucketCors([
        'Bucket' => $bucketName, // REQUIRED
        'CORSConfiguration' => [ // REQUIRED
            'CORSRules' => [ // REQUIRED
                [
                    'AllowedHeaders' => ['Authorization'],
                    'AllowedMethods' => ['POST', 'GET', 'PUT'], // REQUIRED
                    'AllowedOrigins' => ['*'], // REQUIRED
                    'ExposeHeaders' => [],
                    'MaxAgeSeconds' => 3000
                ],
            ],
        ]
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3에서 Amazon S3 멀티파트 업로드 사용

단일 `PutObject` 작업으로 최대 5GB 크기의 객체를 업로드할 수 있습니다. 하지만 멀티파트 업로드 메서드(예: `CreateMultipartUpload`, `UploadPart`, `CompleteMultipartUpload`, `AbortMultipartUpload`)를 사용하면 5MB ~ 5TB 크기의 객체를 업로드할 수 있습니다.

다음 예에서는 작업 방법을 보여줍니다.

- ObjectUploader를 사용하여 Amazon S3에 객체를 업로드합니다.
- MultipartUploader를 사용하여 Amazon S3 객체에 대한 멀티파트 업로드를 생성합니다.
- ObjectCopier를 사용하여 Amazon S3 위치 간 객체를 복사합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

객체 업로더

작업에 PutObject와 MultipartUploader 중 어느 것이 더 적합한지 확실하지 않다면 ObjectUploader를 사용하세요. ObjectUploader는 페이로드 크기에 따라 PutObject 또는 MultipartUploader 중 하나를 사용하여 대용량 파일을 Amazon S3에 업로드합니다.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

샘플 코드

```
// 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);
```

구성

ObjectUploader 객체 생성자는 다음 인수를 받습니다.

$client

전송을 수행하는 데 사용할 Aws\ClientInterface 객체입니다. 이 객체는 Aws\S3\S3Client의 인스턴스여야 합니다.

$bucket

(string, 필수) 객체가 업로드되는 버킷의 이름입니다.

$key

(string, 필수) 업로드되는 객체에 사용할 키입니다.

$body

(mixed, 필수) 업로드할 객체 데이터. StreamInterface일 수 있습니다, PHP 스트림 리소스 또는 업로드할 데이터 문자열일 수 있습니다.

$acl

(string) 업로드되는 객체에서 설정할 ACL(액세스 제어 목록)입니다. 객체는 기본적으로 프라이빗입니다.

$options

멀티파트 업로드에 대한 구성 옵션의 결합형 배열입니다. 유효한 구성 옵션은 다음과 같습니다.

add_content_md5

(bool) true로 설정하면 업로드에 필요한 MD5 체크섬이 자동으로 계산됩니다.

mup_threshold

(int, 기본값: int(16777216)) 파일 크기의 바이트 수입니다. 파일 크기가 이 제한을 초과하는 경우 멀티파트 업로드가 사용됩니다.

before_complete

(callable) CompleteMultipartUpload 작업 이전에 호출할 콜백입니다. 콜백에는 function (Aws\Command $command) {...}와 같은 함수 서명이 있어야 합니다. CommandInterface 객체에 추가할 수 있는 파라미터는 CompleteMultipartUpload API 참조를 참조하세요.

before_initiate

(callable) CreateMultipartUpload 작업 이전에 호출할 콜백입니다. 콜백에는 function (Aws\Command $command) {...}와 같은 함수 서명이 있어야 합니다. 파일 크기가 mup_threshold 값을 초과하면 이 콜백이 호출됩니다. CommandInterface 객체에 추가할 수 있는 파라미터는 CreateMultipartUpload API 참조를 참조하세요.

before_upload

(callable) 모든 PutObject 또는 UploadPart 작업 이전에 호출할 콜백입니다. 콜백에는 function (Aws\Command $command) {...}와 같은 함수 서명이 있어야 합니다. 파일 크기가 mup_threshold 값보다 작거나 같으면 이 콜백이 호출됩니다. PutObject 요청에 적용할 수 있는 파라미터는 PutObject API 참조를 참조하세요. UploadPart 요청에 적용되는 파라미터는 UploadPart API 참조를 참조하세요. CommandInterface 객체로 표시되는 작업에 적용되지 않는 모든 파라미터를 무시됩니다.

concurrency

(int, 기본값: int(3)) 멀티파트 업로드 중에 허용되는 최대 동시 실행 UploadPart 작업 수입니다.

part_size

(int, 기본값: int(5242880)) 멀티파트 업로드를 수행할 때 사용할 부분 크기(바이트)입니다. 값은 5MB에서 5GB 사이(포함)이어야 합니다.

state

(Aws\Multipart\UploadState) 멀티파트 업로드의 상태를 나타내며 이전 업로드를 다시 시작하는 데 사용되는 객체입니다. 이 옵션을 제공하면 $bucket, $key 인수 및 part_size 옵션이 무시됩니다.

params

각 하위 명령에 대한 구성 옵션을 제공하는 결합형 배열입니다. 예:

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

MultipartUploader

멀티파트 업로드는 대용량 객체의 업로드 경험을 개선하기 위해 디자인되었습니다. 이 메서드를 사용하면 객체를 독립적인 부분으로 어떤 순서로든 병렬로 업로드할 수 있습니다.

Amazon S3 고객은 100MB를 초과하는 객체에 멀티파트 업로드를 사용하는 것이 좋습니다.

MultipartUploader 객체

SDK에는 멀티파트 업로드 프로세스를 간소화하는 특수한 `MultipartUploader` 객체가 있습니다.

가져옵니다.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

샘플 코드

```
$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";
}
```

업로더는 제공된 소스와 구성을 기반으로 부분 데이터 생성기를 생성하고 모든 부분을 업로드하려고 시도합니다. 일부 부분 업로드가 실패하면 업로더는 전체 소스 데이터를 읽을 때까지 이후 부분을 계속해서 업로드합니다. 그런 다음 업로더가 실패한 부분을 다시 업로드하거나, 혹은 업로드에 실패한 부분에 대한 정보가 포함된 예외를 발생시킵니다.

멀티파트 업로드 사용자 지정

생성기에 전달된 콜백을 통해 멀티파트 업로더에서 실행되는 `CreateMultipartUpload`, `UploadPart` 및 `CompleteMultipartUpload` 작업에 대해 사용자 지정 옵션을 설정할 수 있습니다.

가져옵니다.

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

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

샘플 코드

```
// 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';
    },
]);
```

부분 업로드 간 수동 가비지 수집

대용량 업로드로 인해 메모리 제한에 도달한 경우에는 메모리 제한에 도달했을 때 PHP 가비지 수집기에서 수집된 순환 참조가 아닌, SDK에서 생성된 순환 참조가 원인일 수 있습니다. 이때는 작업 사이에 수집 알고리즘을 직접 호출하면 제한에 도달하기 전에 순환 참조를 수집할 수 있습니다. 다음 예제는 각 부분 업로드 전에 콜백을 사용해 수집 알고리즘을 호출하는 것입니다. 단, 가비지 수집기를 호출할 경우 성능 비용이 발생하므로 사용 사례와 환경에 따라 사용하는 것이 좋습니다.

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

오류 복구

멀티파트 업로드 프로세스 중에 오류가 발생하면 `MultipartUploadException`이 발생합니다. 이 예외는 멀티파트 업로드의 진행률 정보가 포함된 `UploadState` 객체에 대한 액세스를 제공합니다. `UploadState`를 사용하여 완료하지 못한 업로드를 다시 시작할 수 있습니다.

가져옵니다.

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

use Aws\Exception\MultipartUploadException;
```

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

샘플 코드

```
// 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);
}
```

UploadState에서 업로드를 다시 시작하면 아직 업로드되지 않은 부분을 업로드하려고 시도합니다. 상태 객체는 업로드가 연속적이 아닌 경우 누락된 부분을 계속 추적합니다. 업로더는 제공된 소스 파일을 아직도 업로드해야 하는 부분에 속한 바이트 범위까지 세부적으로 읽거나 탐색합니다.

`UploadState` 객체는 직렬화 가능하므로, 다른 프로세스에서 업로드를 다시 시작할 수도 있습니다. 또한 예외를 처리하지 않을 때에도 `UploadState`를 호출하여 `$uploader->getState()` 객체를 가져올 수 있습니다.

> **Important**
>
> `MultipartUploader`에 소스로 전달된 스트림은 업로드 전에 자동으로 되감지 않습니다. 이전의 예제와 비슷한 루프에서 파일 경로 대신 스트림을 사용하는 경우 `$source` 블록 내부의 `catch` 변수를 재설정합니다.

가져옵니다.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

샘플 코드

```
// 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);
```

멀티파트 업로드 중단

멀티파트 업로드는 UploadId 객체에 포함된 UploadState을 불러와 abortMultipartUpload에 전달하여 중단할 수 있습니다.

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

비동기 멀티파트 업로드

upload()에서 MultipartUploader를 호출하는 것은 차단 요청입니다. 비동기 콘텍스트에서 작업하는 경우 멀티파트 업로드에 대한 promise를 가져올 수 있습니다.

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

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

샘플 코드

```
// 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();
```

구성

`MultipartUploader` 객체 생성자는 다음 인수를 받습니다.

`$client`

전송을 수행하는 데 사용할 `Aws\ClientInterface` 객체입니다. 이 객체는 `Aws\S3\S3Client`의 인스턴스여야 합니다.

`$source`

업로드되는 소스 데이터입니다. 이 데이터는 경로 또는 URL(예: `/path/to/file.jpg`), 리소스 핸들(예: `fopen('/path/to/file.jpg', 'r')`) 또는 PSR-7 스트림의 인스턴스일 수 있습니다.

`$config`

멀티파트 업로드에 대한 구성 옵션의 결합형 배열입니다.

유효한 구성 옵션은 다음과 같습니다.

`acl`

(`string`) 업로드되는 객체에서 설정할 ACL(액세스 제어 목록)입니다. 객체는 기본적으로 프라이빗입니다.

`before_complete`

(`callable`) `CompleteMultipartUpload` 작업 이전에 호출할 콜백입니다. 콜백에는 `function (Aws\Command $command) {...}`와 같은 함수 서명이 있어야 합니다.

`before_initiate`

(`callable`) `CreateMultipartUpload` 작업 이전에 호출할 콜백입니다. 콜백에는 `function (Aws\Command $command) {...}`와 같은 함수 서명이 있어야 합니다.

`before_upload`

(`callable`) 모든 `UploadPart` 작업 이전에 호출할 콜백입니다. 콜백에는 `function (Aws\Command $command) {...}`와 같은 함수 서명이 있어야 합니다.

bucket

(string, 필수) 객체가 업로드되는 버킷의 이름입니다.

concurrency

(int, 기본값: int(5)) 멀티파트 업로드 중에 허용되는 최대 동시 실행 UploadPart 작업 수입니다.

key

(string, 필수) 업로드되는 객체에 사용할 키입니다.

part_size

(int, 기본값: int(5242880)) 멀티파트 업로드를 수행할 때 사용할 부분 크기(바이트)입니다. 이 값은 5MB에서 5GB 사이(포함)이어야 합니다.

state

(Aws\Multipart\UploadState) 멀티파트 업로드의 상태를 나타내며 이전 업로드를 다시 시작하는 데 사용되는 객체입니다. 이 옵션을 제공하면 bucket, key 및 part_size 옵션이 무시됩니다.

add_content_md5

(boolean) true로 설정하면 업로드에 필요한 MD5 체크섬이 자동으로 계산됩니다.

params

각 하위 명령에 대한 구성 옵션을 제공하는 결합형 배열입니다. 예:

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

멀티파트 복사

AWS SDK for PHP는 MultipartUploader와 비슷한 방식으로 사용되는 MultipartCopy 객체를 포함하지만, Amazon S3 내에서 5GB ~ 5TB 크기의 객체를 복사하기 위해 설계되었습니다.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
```

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

샘플 코드

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

//Copy objects within S3
$copier = new MultipartCopy($s3Client, '/bucket/key?versionId=foo', [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

try {
    $result = $copier->copy();
    echo "Copy complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3의 사전 서명된 URL

HTTP 인증 헤더를 사용하는 대신 쿼리 문자열 파라미터로 필요한 정보를 전달하여 특정 유형의 요청을 인증할 수 있습니다. 이 방법은 요청을 프록시하지 않고 타사 브라우저에서 Amazon S3 비공개 데이터에 직접 액세스할 수 있도록 할 경우에 유용합니다. 핵심은 "미리 서명된" 요청을 구성하여 다른 사용자가 사용할 수 있는 URL로 인코딩하는 것입니다. 또한 만료 시간을 지정하여 미리 서명된 요청을 제한할 수 있습니다.

HTTP GET 요청에 대해 미리 서명된 URL 생성

다음 코드 예시는 SDK for PHP를 사용하여 HTTP GET 요청에 대해 미리 서명된 URL을 생성하는 방법을 보여줍니다.

```
<?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;
```

자세한 내용은 `createPresignedRequest` 메서드에 대한 API 참조에서 확인할 수 있습니다.

다른 사용자는 `$presignedUrl` 값을 사용하여 향후 1시간 이내에 해당 객체를 가져올 수 있습니다. 예를 들어 브라우저를 통해 HTTP GET 요청이 수행되면 S3 서비스에는 이 호출이 사전 서명된 URL을 생성한 사용자로부터 온 것으로 인식됩니다.

HTTP PUT 요청에 대해 미리 서명된 URL 생성

다음 코드 예시는 SDK for PHP를 사용하여 HTTP PUT 요청에 대해 미리 서명된 URL을 생성하는 방법을 보여줍니다.

```
<?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();
```

이제 다른 사용자가 미리 서명된 URL을 HTTP PUT 요청에 사용하여 파일을 업로드할 수 있습니다.

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

// ...

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

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

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

    // Send the request using the handler.
    try {
        $promise = $handler($request, []);
        $response = $promise->wait();
        return $response;
    } catch (Exception $e) {
```

```
        echo "Error uploading file: " . $e->getMessage() . "\n";
        return null;
    }
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3에서 사전 서명된 게시물

미리 서명된 URL과 마찬가지로 미리 서명된 POST를 사용하여 AWS 보안 인증을 제공하지 않고 사용자에게 쓰기 권한을 부여할 수 있습니다. AwsS3PostObjectV4의 인스턴스를 활용하여 미리 서명된 POST 양식을 생성할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- PostObjectV4를 사용해 S3 Object POST 업로드 형식에 적합한 데이터를 가져옵니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

> **Note**
>
> PostObjectV4에서 AWS IAM Identity Center 가져온 보안 인증으로는 작동하지 않습니다.

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에서 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

PostObjectV4 생성

PostObjectV4 인스턴스를 생성하려면 다음을 제공해야 합니다.

- Aws\S3\S3Client 인스턴스
- 버킷
- 양식 입력 필드의 결합형 배열
- 정책 조건 배열 (Amazon Simple Storage Service 사용 설명서의 정책 구성 참조)
- 정책에 대한 만료 시간 문자열(선택 사항, 기본적으로 1시간)

가져옵니다.

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

샘플 코드

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;

$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-1',
]);
$bucket = 'amzn-s3-demo-bucket10';
$starts_with = 'user/eric/';
$client->listBuckets();

// Set defaults for form input fields.
$formInputs = ['acl' => 'public-read'];

// Construct an array of conditions for policy.
$options = [
    ['acl' => 'public-read'],
    ['bucket' => $bucket],
    ['starts-with', '$key', $starts_with],
];

// Set an expiration time (optional).
$expires = '+2 hours';

$postObject = new PostObjectV4(
    $client,
    $bucket,
    $formInputs,
    $options,
    $expires
);

// Get attributes for the HTML form, for example, action, method, enctype.
$formAttributes = $postObject->getFormAttributes();
```

```
// Get attributes for the HTML form values.
$formInputs = $postObject->getFormInputs();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>PHP</title>
</head>
<body>
<form action="<?php echo $formAttributes['action'] ?>" method="<?php echo
 $formAttributes['method'] ?>"
      enctype="<?php echo $formAttributes['enctype'] ?>">
    <label id="key">
        <input hidden type="text" name="key" value="<?php echo $starts_with ?><?php
 echo $formInputs['key'] ?>"/>
    </label>
    <h3>$formInputs:</h3>
    acl: <label id="acl">
        <input readonly type="text" name="acl" value="<?php echo $formInputs['acl'] ?
>"/>
    </label><br/>
    X-Amz-Credential: <label id="credential">
        <input readonly type="text" name="X-Amz-Credential" value="<?php echo
 $formInputs['X-Amz-Credential'] ?>"/>
    </label><br/>
    X-Amz-Algorithm: <label id="algorithm">
        <input readonly type="text" name="X-Amz-Algorithm" value="<?php echo
 $formInputs['X-Amz-Algorithm'] ?>"/>
    </label><br/>
    X-Amz-Date: <label id="date">
        <input readonly type="text" name="X-Amz-Date" value="<?php echo $formInputs['X-
Amz-Date'] ?>"/>
    </label><br/><br/><br/>
    Policy: <label id="policy">
        <input readonly type="text" name="Policy" value="<?php echo
 $formInputs['Policy'] ?>"/>
    </label><br/>
    X-Amz-Signature: <label id="signature">
        <input readonly type="text" name="X-Amz-Signature" value="<?php echo
 $formInputs['X-Amz-Signature'] ?>"/>
    </label><br/><br/>
    <h3>Choose file:</h3>
```

```
    <input type="file" name="file"/> <br/><br/>
    <h3>Upload file:</h3>
    <input type="submit" name="submit" value="Upload to Amazon S3"/>
</form>
</body>
</html>
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷을 정적 웹 호스트로 사용

Amazon S3에 정적 웹 사이트를 호스팅할 수 있습니다. 자세한 내용은 Amazon S3 정적 웹 사이트 호스팅을 참조하세요.

다음 예에서는 작업 방법을 보여줍니다.

- GetBucketWebsite를 사용하여 버킷에 대한 웹 사이트 구성을 가져옵니다.
- PutBucketWebsite를 사용하여 버킷에 대한 웹 사이트 구성을 설정합니다.
- DeleteBucketWebsite를 사용하여 버킷에서 웹 사이트 구성을 제거합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 AWS 보안 인증을 구성합니다. AWS SDK for PHP 버전 3의 보안 인증을 참조하세요.

버킷에 대한 웹 사이트 구성 가져오기, 설정, 삭제

가져옵니다.

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

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

샘플 코드

```
$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";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon S3 버킷 정책 작업

버킷 정책을 사용하여 Amazon S3 리소스에 대한 권한을 부여할 수 있습니다. 자세한 내용은 버킷 정책 및 사용자 정책 사용을 참조하세요.

다음 예에서는 작업 방법을 보여줍니다.

- GetBucketPolicy를 사용하여 지정된 버킷에 대한 정책을 반환합니다.
- PutBucketPolicy를 사용하여 버킷에 대한 정책을 대체합니다.
- DeleteBucketPolicy를 사용하여 버킷에서 정책을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

버킷에 대한 정책 가져오기, 삭제, 바꾸기

가져옵니다.

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

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

샘플 코드

```
$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";
}
```

S3 액세스 포인트 ARN AWS SDK for PHP 버전 3사용

S3는 S3 버킷과 상호 작용하는 새로운 방법인 액세스 포인트를 도입했습니다. 액세스 포인트는 버킷에 직접 적용되는 대신 고유한 정책 및 구성을 적용할 수 있습니다. AWS SDK for PHP를 사용하면 버킷 이름을 명시적으로 지정하는 대신 API 작업에 버킷 필드에서 액세스 포인트 ARN을 사용할 수 있습니다. 여기에서 S3 액세스 포인트 및 ARN의 작동 방식에 대한 자세한 내용을 확인할 수 있습니다. 다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- 액세스 포인트 ARN과 함께 GetObject를 사용하여 버킷에서 객체를 가져옵니다.
- 액세스 포인트 ARN과 함께 PutObject를 사용하여 버킷에 객체를 추가합니다.
- 클라이언트 리전 대신 ARN 리전을 사용하도록 S3 클라이언트를 구성합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

가져옵니다.

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

use Aws\S3\S3Client;
```

객체 가져오기

먼저 AWS 리전과 버전을 지정하는 AWS.S3 클라이언트 서비스를 생성합니다. 그런 다음 `getObject` 필드에서 키와 S3 액세스 포인트 ARN을 사용하여 `Bucket` 메서드를 호출하면 해당 액세스 포인트와 연결된 버킷에서 객체를 가져옵니다.

샘플 코드

```
$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'
]);
```

버킷에 객체 넣기

먼저 AWS 리전과 버전을 지정하는 AWS.S3 클라이언트 서비스를 생성합니다. 그런 다음 putObject 필드에서 원하는 키, 본문 또는 소스 파일과 S3 액세스 포인트 ARN을 사용하여 Bucket 메서드를 호출하면 해당 액세스 포인트와 연결된 버킷에 객체가 배치됩니다.

샘플 코드

```
$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'
]);
```

클라이언트 리전 대신 ARN 리전을 사용하도록 S3 클라이언트를 구성합니다.

S3 클라이언트 작업에서 S3 액세스 포인트 ARN을 사용하는 경우 기본적으로 클라이언트는 ARN 리전이 클라이언트 리전과 일치하는지 확인하고 그렇지 않은 경우 예외가 발생합니다. 이 동작은 use_arn_region 구성 옵션을 true로 설정하여 클라이언트 리전에서 ARN 리전을 수락하도록 변경할 수 있습니다. 기본적으로 이 옵션은 false로 설정되어 있습니다.

샘플 코드

```
$s3 = new S3Client([
    'version'        => 'latest',
    'region'         => 'us-west-2',
    'use_arn_region' => true
]);
```

클라이언트는 환경 변수와 구성 파일 옵션을 다음과 같은 우선 순위에 따라 확인합니다.

1. 위의 예제에서와 같은 클라이언트 옵션 use_arn_region.
2. 환경 변수 AWS_S3_USE_ARN_REGION

```
export AWS_S3_USE_ARN_REGION=true
```

1. AWS 공유 구성 파일(기본적으로 `~/.aws/config`)의 구성 변수 `s3_use_arn_region`.

```
[default]
s3_use_arn_region = true
```

AWS SDK for PHP 버전 3에서 Amazon S3 다중 리전 액세스 포인트 사용

Amazon Simple Storage Service(S3) 다중 리전 액세스 포인트는 Amazon S3 요청 트래픽을 AWS 리전간에 라우팅하기 위한 글로벌 엔드포인트를 제공합니다.

SDK for PHP, 다른 SDK, S3 콘솔 또는 CLI를 사용하여 다중 리전 액세스 포인트를 생성할 수 있습니다. AWS S3 AWS

> **Important**
>
> PHP용 SDK에서 다중 리전 액세스 포인트를 사용하려면 PHP 환경에 AWS 공통 런타임(AWS CRT) 확장이 설치되어 있어야 합니다.

다중 리전 액세스 포인트를 생성하면 Amazon S3에서는 다음 형식의 Amazon 리소스 이름(ARN)을 생성합니다.

`arn:aws:s3::`*`account-id`*`:accesspoint/`*`MultiRegionAccessPoint_alias`*

생성된 ARN을 `getObject()` 및 `putObject()` 메서드의 버킷 이름 대신 사용할 수 있습니다.

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

use Aws\S3\S3Client;

// Assign the Multi-Region Access Point to a variable and use it place of a bucket
 name.
$mrap = 'arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap';
$key = 'my-key';

$s3Client = new S3Client([
    'region' => 'us-east-1'
```

```
]);

$s3Client->putObject([
    'Bucket' => $mrap,
    'Key' => $key,
    'Body' => 'Hello World!'
]);

$result = $s3Client->getObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

echo $result['Body'] . "\n";

// Clean up.
$result = $s3Client->deleteObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

$s3Client->waitUntil('ObjectNotExists', ['Bucket' => $mrap, 'Key' => $key]);

echo "Object deleted\n";
```

Secrets Manager API 및 AWS SDK for PHP 버전 3을 사용한 암호 관리

AWS Secrets Manager는 암호, API 키, 데이터베이스 보안 인증 같은 공유 암호을 저장하고 관리합니다. 개발자는 이러한 Secrets Manager 서비스에서 Secrets Manager 호출 기능을 사용해 배포 코드에서 하드 코딩된 보안 인증을 변경할 수 있습니다.

Secrets Manager는 Amazon Relational Database Service (Amazon RDS) 데이터베이스의 자동 예약 보안 인증 교체를 기본적으로 지원하여 애플리케이션 보안을 강화합니다. 또한 Secrets Manager는 서비스별 세부 정보를 구현하는 데 AWS Lambda를 사용하여 다른 데이터베이스 및 타사 서비스에 대한 보안 정보를 원활하게 교체할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateSecret을 사용해 암호를 생성합니다.
- GetSecretValue를 사용해 암호를 가져옵니다.
- ListSecrets를 사용해 Secrets Manager에서 저장된 암호를 모두 나열합니다.

- DescribeSecret을 사용해 특정 암호에 대한 세부 정보를 가져옵니다.
- PutSecretValue를 사용해 특정 암호를 업데이트합니다.
- RotateSecret을 사용해 암호 교체를 설정합니다.
- DeleteSecret을 사용해 삭제할 암호를 표시합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

Secrets Manager에서 보안 암호 생성

Secrets Manager에서 암호를 생성할 때는 CreateSecret 작업을 사용합니다.

이번 예제에서는 사용자 이름과 암호가 JSON 문자열로 저장되어 있습니다.

가져옵니다.

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
$description = '<<Description>>';
try {
    $result = $client->createSecret([
        'Description' => $description,
        'Name' => $secretName,
```

```
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager에서 문자열 보안 암호 검색

Secrets Manager에 저장된 암호 값을 가져올 때는 GetSecretValue 작업을 사용합니다.

이번 예제에서 `secret`는 저장된 값이 포함된 문자열입니다. `username`의 값이 `<<USERNAME>>`이고 `<<PASSWORD>>`의 값이 `password`인 경우 `secret` 출력은 다음과 같습니다.

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

배열 값에 액세스하는 데 `json_decode($secret, true)`를 사용합니다.

가져옵니다.

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-east-1',
]);

$secretName = 'MySecretName';

try {
    $result = $client->getSecretValue([
        'SecretId' => $secretName,
```

```
    ]);
} catch (AwsException $e) {
    $error = $e->getAwsErrorCode();
    if ($error == 'DecryptionFailureException') {
        // Secrets Manager can't decrypt the protected secret text using the provided
 AWS KMS key.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InternalServiceErrorException') {
        // An error occurred on the server side.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidParameterException') {
        // You provided an invalid value for a parameter.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidRequestException') {
        // You provided a parameter value that is not valid for the current state of
 the resource.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'ResourceNotFoundException') {
        // We can't find the resource that you asked for.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
}
// Decrypts secret using the associated KMS CMK.
// Depending on whether the secret is a string or binary, one of these fields will be
 populated.
if (isset($result['SecretString'])) {
    $secret = $result['SecretString'];
} else {
    $secret = base64_decode($result['SecretBinary']);
}
print $secret;
$secretArray = json_decode($secret, true);
$username = $secretArray['username'];
$password = $secretArray['password'];
```

```
// Your code goes here;
```

Secrets Manager에 저장된 보안 암호 생성

Secrets Manager에서 저장된 암호 목록을 모두 나열할 때는 ListSecrets 작업을 사용합니다.

가져옵니다.

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

암호에 대한 세부 정보 가져오기

저장된 암호에는 교체 규칙에 대한 메타데이터, 마지막 액세스 또는 변경 시점, 사용자 생성 태그, Amazon 리소스 이름(ARN)이 포함되어 있습니다. Secrets Manager에 저장된 특정 암호에 대한 세부 정보를 가져올 때는 DescribeSecret 작업을 사용합니다.

가져옵니다.

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

```
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

암호 값 업데이트

새롭게 암호화된 암호 값을 Secrets Manager에 저장할 때는 PutSecretValue 작업을 사용합니다.

그러면 새로운 버전의 암호가 생성됩니다. 암호 버전이 이미 존재한다면 값과 함께 `VersionStages` 파라미터를 `AWSCURRENT`에 추가하여 값을 가져올 때 새로운 값이 사용되도록 합니다.

가져옵니다.

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new SecretsManagerClient([
    'profile' => 'default',
```

```
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
try {
    $result = $client->putSecretValue([
        'SecretId' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager에 저장된 기존 암호 값 교체

Secrets Manager에 저장된 기존 암호 값을 교체할 때는 Ladmbda 교체 함수와 RotateSecret 작업을 사용합니다.

시작하기 전에 먼저 암호을 교체할 때 사용할 Lambda 함수를 생성합니다. 현재 AWS 코드 샘플 카탈로그에는 Amazon RDS 데이터베이스 보안 인증을 교체할 때 사용되는 Lambda 코드 예제가 몇 가지 포함되어 있습니다.

> **Note**
>
> 자세한 내용은 AWS Secrets Manager 사용 설명서에서 AWS Secrets Manager 암호 교체를 참조하세요.

Lambda 함수를 설정하였으면 이제 새로운 암호 교체를 구성합니다.

가져옵니다.

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

use Aws\SecretsManager\SecretsManagerClient;
```

```
use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

교체가 구성되면 이제 RotateSecret 작업을 사용해 교체를 실행할 수 있습니다.

가져옵니다.

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new SecretsManagerClient([
```

```
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);

$secretName = 'MySecretName';

try {
    $result = $client->rotateSecret([
        'SecretId' => $secretName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Secrets Manager에서 보안 암호 삭제

Secrets Manager에서 지정된 암호을 제거할 때는 DeleteSecret 작업을 사용합니다. 암호을 우발적으로 삭제하지 못하도록 DeletionDate 스탬프가 암호에 자동으로 추가되어 삭제를 되돌릴 수 있는 복구 시간을 지정할 수 잇습니다. 복구 시간을 지정하지 않으면 기본 시간으로 30일이 지정됩니다.

가져옵니다.

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

관련 정보

AWS SDK for PHP 예제는 AWS Secrets Manager API 참조에서 다음 REST 작업을 사용합니다.

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

AWS Secrets Manager 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하세요.

AWS SDK for PHP 버전 3을 사용한 Amazon SES 예제

Amazon Simple Email Service(Amazon SES)는 사용자의 이메일 주소와 도메인을 사용해 이메일을 보내고 받기 위한 비용을 절약할 수 있고 사용이 손쉬운 이메일 플랫폼입니다. Amazon SES에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

AWS는 두 가지 버전의 Amazon SES 서비스를 제공하며, 이에 따라 PHP용 SDK는 SESClient와 SesV2Client라는 두 가지 버전의 클라이언트를 제공합니다. 메서드가 호출되는 방식이나 결과가 다를 수 있지만 클라이언트의 기능이 중복되는 경우가 많습니다. 두 API는 독점적인 기능도 제공하므로 두 클라이언트를 모두 사용하여 모든 기능에 액세스할 수 있습니다.

이 단원의 예제에서는 모두 원본 `SesClient`를 사용합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 보안 인증 확인

Amazon Simple Email Service (Amazon SES) 계정 사용을 처음 시작할 때 이메일을 보내려는 AWS 리전과 동일한 리전에서 모든 발신자와 수신자를 확인해야 합니다. 이메일 전송에 대한 자세한 내용은 Amazon SES를 통해 이메일 전송을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- VerifyEmailIdentity를 사용하여 이메일 주소 확인
- VerifyDomainIdentity를 사용하여 이메일 도메인 확인
- ListIdentities를 사용하여 모든 이메일 주소 나열
- ListIdentities를 사용하여 모든 이메일 도메인 나열
- DeleteIdentity를 사용하여 이메일 주소 제거
- DeleteIdentity를 사용하여 이메일 도메인 제거

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

이메일 주소 확인

Amazon SES는 확인된 이메일 주소 또는 도메인에서만 이메일을 보낼 수 있습니다. 이메일 주소를 확인하여 해당 주소의 소유자이고 Amazon SES가 해당 주소에서 이메일을 보낼 수 있도록 하고자 함을 입증합니다.

다음 코드 예제를 실행하면 Amazon SES가 지정한 주소로 이메일을 보냅니다. 사용자(또는 이메일 수신자)가 이메일의 링크를 클릭하면 주소가 확인됩니다.

Amazon SES 계정에 이메일 주소를 추가하려면 VerifyEmailIdentity 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 도메인 확인

Amazon SES는 확인된 이메일 주소 또는 도메인에서만 이메일을 보낼 수 있습니다. 도메인을 확인하여 해당 도메인의 소유자임을 입증합니다. 도메인을 확인하면 Amazon SES가 해당 도메인의 어떤 주소에서든 이메일을 보낼 수 있습니다.

다음 코드 예제를 실행하면 Amazon SES가 확인 토큰을 제공합니다. 도메인의 DNS 구성에 토큰을 추가해야 합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES에서 도메인 확인을 참조하세요.

Amazon SES 계정에 전송 도메인을 추가하려면 VerifyDomainIdentity 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 주소 나열

확인 상태와 관계없이 현재 AWS 리전에 제출된 이메일 주소 목록을 검색하려면 ListIdentities 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 도메인 나열

확인 상태와 관계없이 현재 AWS 리전에 제출된 이메일 도메인 목록을 검색하려면 ListIdentities 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 주소 삭제

보안 인증 목록에서 확인된 이메일 주소를 삭제하려면 DeleteIdentity 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 도메인 삭제

확인된 보안 인증 목록에서 확인된 이메일 도메인을 삭제하려면 DeleteIdentity 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$domain = 'domain.name';

try {
    $result = $SesClient->deleteIdentity([
        'Identity' => $domain,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 사용자 지정 이메일 템플릿 생성

Amazon Simple Email Service (Amazon SES)를 통해 템플릿을 사용하여 각 수신자에 대해 맞춤화된 이메일을 전송할 수 있습니다. 템플릿에는 제목 줄과 이메일 본문의 텍스트 및 HTML 부분이 포함됩니다. 제목과 본문 섹션에는 각 수신자에 대해 맞춤화된 고유 값이 포함될 수도 있습니다.

자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 Amazon SES를 사용하여 개인화된 이메일 전송을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateTemplate을 사용하여 이메일 템플릿 만들기
- ListTemplates를 사용하여 모든 이메일 템플릿 나열
- GetTemplate을 사용하여 이메일 템플릿 검색
- UpdateTemplate을 사용하여 이메일 템플릿 업데이트
- DeleteTemplate을 사용하여 이메일 템플릿 제거
- SendTemplatedEmail을 사용하여 템플릿 이메일 전송

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

이메일 템플릿 생성

템플릿을 만들어 맞춤형 이메일 메시지를 전송하려면 CreateTemplate 작업을 사용합니다. 템플릿은 해당 템플릿이 추가된 AWS 리전에서 메시지를 보낼 권한이 있는 계정이 사용할 수 있습니다.

> **Note**
>
> Amazon SES는 HTML의 유효성을 검사하지 않으므로 이메일을 전송하기 전에 HtmlPart가 유효한지 확인해야 합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 템플릿 가져오기

제목 줄, HTML 본문 및 일반 텍스트를 포함하여 기존 이메일 템플릿의 콘텐츠를 보려면 GetTemplate 작업을 사용합니다. TemplateName만 필수입니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

모든 이메일 템플릿 나열

현재 AWS 리전의 AWS 계정 계정과 연결된 모든 이메일 템플릿의 목록을 검색하려면 ListTemplates 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 템플릿 업데이트

제목 줄, HTML 본문 및 일반 텍스트를 포함하여 특정 이메일 템플릿의 콘텐츠를 변경하려면 UpdateTemplate 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 템플릿 삭제

특정 이메일 템플릿을 제거하려면 DeleteTemplate 작업을 사용합니다. TemplateName만 필요합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

템플릿이 있는 이메일 전송

템플릿을 사용하여 수신자에게 이메일을 전송하려면 SendTemplatedEmail 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$template_name = 'Template_Name';
$sender_email = 'email_address';
```

```
$recipient_emails = ['email_address'];

try {
    $result = $SesClient->sendTemplatedEmail([
        'Destination' => [
            'ToAddresses' => $recipient_emails,
        ],
        'ReplyToAddresses' => [$sender_email],
        'Source' => $sender_email,

        'Template' => $template_name,
        'TemplateData' => '{ }'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 필터 관리

이메일을 보내는 것 외에도 Amazon Simple Email Service (Amazon SES) 로 이메일을 수신할 수 있습니다. IP 주소 필터는 특정 IP 주소 또는 특정 범위의 IP 주소에서 발신한 메일의 수락 여부를 지정할 수 있게 합니다. 자세한 내용은 Amazon SES 이메일 수신을 위해 IP 주소 필터 관리 단원을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateReceiptFilter를 사용하여 이메일 필터 만들기
- ListReceiptFilters를 사용하여 모든 이메일 필터 나열
- DeleteReceiptFilter를 사용하여 이메일 필터 제거

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

이메일 필터 만들기

특정 IP 주소에서 보내는 이메일을 허용하거나 차단하려면 CreateReceiptFilter 작업을 사용합니다. IP 주소 또는 주소 범위와 이 필터를 식별할 수 있는 고유한 이름을 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

모든 이메일 필터 나열

현재 AWS 리전의 AWS 계정과 연결된 IP 주소 필터를 나열하려면 ListReceiptFilters 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

이메일 필터 삭제

특정 IP 주소에 대한 기존 필터를 제거하려면 DeleteReceiptFilter 작업을 사용합니다. 삭제할 수신 필터를 식별할 수 있는 고유한 필터 이름을 제공합니다.

필터링되는 주소 범위를 변경해야 하는 경우, 수신 필터를 삭제하고 새로 만들 수 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$filter_name = 'FilterName';

try {
    $result = $SesClient->deleteReceiptFilter([
        'FilterName' => $filter_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 규칙 생성 및 관리

이메일을 보내는 것 외에도 Amazon Simple Email Service (Amazon SES)를 사용하여 이메일을 수신할 수 있습니다. 수신 규칙을 사용하면 사용자 소유의 이메일 주소 또는 도메인으로 수신되는 이메일에 대한 Amazon SES의 처리 방법을 지정할 수 있습니다. 규칙은 Amazon S3, Amazon SNS, AWS Lambda 등 다른 AWS 서비스로 이메일을 보낼 수 있습니다.

자세한 내용은 Amazon SES 이메일 수신을 위한 수신 규칙 세트 관리하기 및 Amazon SES 이메일 수신을 위한 수신 규칙 관리하기를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateReceiptRuleSet를 사용하여 수신 규칙 세트 만들기
- CreateReceiptRule을 사용하여 수신 규칙 만들기

- DescribeReceiptRuleSet를 사용하여 수신 규칙 세트 설명
- DescribeReceiptRule을 사용하여 수신 규칙 설명
- ListReceiptRuleSets를 사용하여 모든 수신 규칙 세트 나열
- UpdateReceiptRule을 사용하여 수신 규칙 업데이트
- DeleteReceiptRule을 사용하여 수신 규칙 제거
- DeleteReceiptRuleSet를 사용하여 수신 규칙 세트 제거

AWS SDK for PHP 에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

수신 규칙 세트 생성

수신 규칙 세트에는 수신 규칙 모음이 포함되어 있습니다. 수신 규칙을 만들려면 계정과 연결된 수신 규칙 세트가 하나 이상 있어야 합니다. 수신 규칙 세트를 만들려면 고유한 RuleSetName을 제공하고 CreateReceiptRuleSet 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 생성

기존의 수신 규칙 세트에 수신 규칙을 추가하여 수신 이메일을 제어합니다. 다음 예제에서는 Amazon S3 버킷에 수신 메시지를 보내는 수신 규칙을 만드는 방법을 보여 주지만, Amazon SNS 및 AWS Lambda에 메시지를 보낼 수도 있습니다. 수신 규칙을 만들려면 규칙과 RuleSetName을 CreateReceiptRule 작업에 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 세트 설명

초당 한 번 지정된 수신 규칙 세트의 세부 정보를 반환합니다. DescribeReceiptRuleSet 작업을 사용하려면 RuleSetName을 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 설명

지정된 수신 규칙의 세부 정보를 반환합니다. DescribeReceiptRule 작업을 사용하려면 RuleName과 RuleSetName을 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

모든 수신 규칙 세트 나열

현재 AWS 리전의 AWS 계정 계정 아래에 있는 수신 규칙 세트를 나열하려면 ListReceiptRuleSets 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 업데이트

다음 예제에서는 AWS Lambda 함수에 수신 메시지를 보내는 수신 규칙을 업데이트하는 방법을 보여 주지만, Amazon SNS 및 Amazon S3에 메시지를 보낼 수도 있습니다. UpdateReceiptRule 작업을 사용하려면 새 수신 규칙과 RuleSetName을 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 세트 삭제

현재 비활성화되지 않은 지정된 수신 규칙 세트를 제거합니다. 그러면 수신 규칙 세트에 포함된 모든 수신 규칙도 삭제됩니다. 수신 규칙 세트를 삭제하려면 RuleSetName을 DeleteReceiptRuleSet 작업에 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$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";
}
```

수신 규칙 삭제

지정된 수신 규칙을 삭제하려면 RuleName과 RuleSetName을 DeleteReceiptRule 작업에 제공합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
```

샘플 코드

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$rule_name = 'Rule_Name';
$rule_set_name = 'Rule_Set_Name';

try {
    $result = $SesClient->deleteReceiptRule([
        'RuleName' => $rule_name,
        'RuleSetName' => $rule_set_name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 발신 활동 모니터링

Amazon Simple Email Service (Amazon SES)는 발신 활동을 모니터링하는 방법을 제공합니다. 이러한 방법을 사용하여 계정의 반송, 수신 거부 및 거부 발생률 같은 주요 지표를 추적하는 것이 좋습니다.

반송 메일 및 수신 거부 발생률이 지나치게 높으면 Amazon SES를 사용하여 이메일을 전송하는 데 어려움을 겪을 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- GetSendQuota를 사용하여 발신 할당량 확인
- GetSendStatistics를 사용하여 전송 활동 모니터링

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

발신 할당량 확인

단일 24시간 동안 특정 양의 메시지만 전송하도록 제한됩니다. 전송할 수 있는 메시지 수를 확인하려면 GetSendQuota 작업을 사용합니다. 자세한 내용은 Amazon SES 발신 한도 관리를 참조하세요.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$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";
}
```

전송 활동 모니터링

지난 2주간 전송한 메시지에 대한 지표를 검색하려면 GetSendStatistics 작업을 사용합니다. 다음 예제에서는 15분 단위로 전송 시도, 반송, 수신 거부 및 거부된 메시지 수를 반환합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

try {
    $result = $SesClient->getSendStatistics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 발신자 권한 부여

다른 AWS 계정, AWS Identity and Access Management 사용자 또는 AWS 서비스가 사용자를 대신해 Amazon Simple Email Service (Amazon SES)를 통해 이메일을 전송할 수 있도록 하려면 전송 권한 부여 정책을 만듭니다. 이는 소유한 보안 인증에 연결하는 JSON 문서입니다.

이 정책은 누가 어떤 조건으로 해당 보안 인증을 사용하여 전송할 수 있는지 명시적으로 나열합니다. 사용자와 사용자가 정책에서 명시적으로 권한을 부여한 개체 이외의 모든 발신자는 이메일을 전송할 수 없습니다. 보안 인증은 연결된 정책이 없을 수도, 하나 또는 여러 개일 수도 있습니다. 또한 다중 정책의 효과를 구현하기 위해 복수의 문을 포함한 단일 정책을 생성할 수도 있습니다.

자세한 내용은 Amazon SES에서 전송 권한 부여 사용을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- PutIdentityPolicy를 사용하여 권한 있는 발신자 만들기
- GetIdentityPolicies를 사용하여 권한 있는 발신자에 대한 정책 검색
- ListIdentityPolicies를 사용하여 권한 있는 발신자 나열
- DeleteIdentityPolicy를 사용하여 권한 있는 발신자의 권한 취소

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하세요.

권한 있는 발신자 만들기

다른 AWS 계정 계정에 사용자를 대신해 이메일을 전송할 수 있는 권한을 부여하려면 보안 인증 정책을 사용하거나 확인된 이메일 주소 또는 도메인에서 이메일을 전송하도록 권한 부여를 추가하거나 업데이트합니다. 보안 인증 정책을 만들려면 PutIdentityPolicy 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$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";
}
```

권한 있는 발신자에 대한 정책 검색

특정 이메일 보안 인증 또는 도메인 보안 인증과 연결된 전송 권한 부여 정책을 반환합니다. 지정된 이메일 주소 또는 도메인에 대한 전송 권한 부여를 가져오려면 GetIdentityPolicy 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$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";
}
```

권한 있는 발신자 나열

현재 AWSS 리전의 특정 이메일 보안 인증 또는 도메인 보안 인증과 연결된 전송 권한 부여 정책을 나열하려면 ListIdentityPolicies 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$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";
}
```

권한 있는 발신자의 권한 취소

DeleteIdentityPolicy 작업으로 연결된 보안 인증 정책을 삭제하여 이메일 보안 인증 또는 도메인 보안 인증으로 이메일을 전송할 수 있는 다른 AWS 계정 계정의 전송 권한 부여를 제거합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

샘플 코드

```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$identity = "arn:aws:ses:us-east-1:123456789012:identity/example.com";
$name = "policyName";

try {
    $result = $SesClient->deleteIdentityPolicy([
        'Identity' => $identity,
        'PolicyName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon SNS 예제

Amazon Simple Notification Service(Amazon SNS)는 구독 중인 엔드포인트 또는 클라이언트에 대한 메시지 전달 또는 전송을 조정 및 관리하는 웹 서비스입니다.

Amazon SNS에는 게시자(생산자라고도 함)와 구독자(소비자라고도 함)라는 두 유형의 클라이언트가 있습니다. 게시자는 주제에 대한 메시지를 생산 및 발송함으로써 구독자와 비동시적으로 통신하는 논리적 액세스 및 커뮤니케이션 채널입니다. 구독자(웹 서버, 이메일 주소, Amazon SQS 대기열, AWS Lambda 함수)는 주제를 구독하는 경우 지원되는 프로토콜(Amazon SQS, HTTP/HTTPS URL, 이메일, AWS SMS, Lambda) 중 하나를 거쳐 메시지 또는 알림을 소비 또는 수신합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- AWS SDK for PHP 버전 3을 사용한 Amazon SNS에서의 주제 관리
- AWS SDK for PHP 버전 3을 사용한 Amazon SNS에서의 구독 관리
- AWS SDK for PHP 버전 3을 사용하여 Amazon SNS에서 SMS 메시지 보내기

AWS SDK for PHP 버전 3을 사용한 Amazon SNS에서의 주제 관리

Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS URL, 이메일, AWS SMS 또는 AWS Lambda에 알림을 보내려면 먼저 해당 주제의 구독자에 대한 메시지 전달을 관리하는 주제를 생성해야 합니다.

관찰자 설계 패턴 측면에서 주제는 제목과 같습니다. 주제를 생성한 후 주제에 메시지가 게시되면 자동으로 알림을 받을 구독자를 추가합니다.

AWS SDK for PHP 버전 3으로 Amazon SNS에서 구독 관리에서 주제 구독에 대해 자세히 알아보세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateTopic을 사용해 주제를 생성하여 알림 게시
- ListTopics를 사용하여 요청자의 주제 목록 반환
- DeleteTopic을 사용하여 주제와 해당 주제의 모든 구독자 삭제
- GetTopicAttributes를 사용하여 주제의 모든 속성 반환
- SetTopicAttributes를 사용하여 주제 소유자가 주제의 속성을 새 값으로 설정할 수 있도록 허용

Amazon SNS 사용에 대한 자세한 내용은 메시지 전송 상태를 위한 Amazon SNS 주제 속성을 참조하세요.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

주제 생성

주제를 생성하려면 CreateTopic 작업을 사용합니다.

AWS 계정 계정의 각 주제 이름은 고유해야 합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제 나열

현재 AWS 리전에서 최대 100개의 기존 주제를 나열하려면 ListTopics 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제 삭제

기존 주제와 주제의 모든 구독을 제거하려면 DeleteTopic 작업을 사용합니다.

구독자에게 아직 전송되지 않은 메시지도 삭제됩니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제 속성 가져오기

단일 기존 주제의 속성을 가져오려면 GetTopicAttributes 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제 속성 설정

단일 기존 주제의 속성을 업데이트하려면 SetTopicAttributes 작업을 사용합니다.

Policy, DisplayName 및 DeliveryPolicy 속성만 설정할 수 있습니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon SNS에서의 구독 관리

Amazon Simple Notification Service (Amazon SNS) 주제를 사용하여 Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, 이메일 주소, AWS Server Migration Service(AWS SMS) 또는 AWS Lambda로 알림을 보낼 수 있습니다.

구독은 구독자에 대한 메시지 전송을 관리하는 주제에 연결되어 있습니다. 주제 생성에 대한 자세한 내용은 AWS SDK for PHP 버전 3으로 Amazon SNS 주제 관리 단원을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- Subscribe를 사용하여 기존 주제 구독
- ConfirmSubscription을 사용하여 구독 확인
- ListSubscriptionsByTopic을 사용하여 기존 구독 나열
- Unsubscribe를 사용하여 구독 삭제
- Publish를 사용하여 주제의 모든 구독자에게 메시지 전송

Amazon SNS 사용에 대한 자세한 내용은 System-to-System Messaging을 위해 Amazon SNS 사용하기 단원을 참조하세요.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

이메일 주소로 주제 구독

이메일 주소의 구독을 시작하려면 Subscribe 작업을 사용합니다.

전달되는 파라미터에 사용되는 값에 따라 subscribe 메서드를 사용하여 서로 다른 여러 엔드포인트가 Amazon SNS 주제를 구독하게 할 수 있습니다. 이는 이 주제의 다른 예제에서 볼 수 있습니다.

다음 예제에서는 엔드포인트가 이메일 주소입니다. 확인 토큰이 이 이메일로 전송됩니다. 수신한 후 3일 이내에 이 확인 토큰을 사용하여 구독을 확인합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

애플리케이션 엔드포인트가 주제를 구독하도록 등록

웹 앱의 구독을 시작하려면 Subscribe 작업을 사용합니다.

전달되는 파라미터에 사용되는 값에 따라 subscribe 메서드를 사용하여 서로 다른 여러 엔드포인트가 Amazon SNS 주제를 구독하게 할 수 있습니다. 이는 이 주제의 다른 예제에서 볼 수 있습니다.

다음 예제에서는 엔드포인트가 URL입니다. 확인 토큰이 이 웹 주소로 전송됩니다. 수신한 후 3일 이내에 이 확인 토큰을 사용하여 구독을 확인합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Lambda 함수로 주제 구독

Lambda 함수의 구독을 시작하려면 Subscribe 작업을 사용합니다.

전달되는 파라미터에 사용되는 값에 따라 subscribe 메서드를 사용하여 서로 다른 여러 엔드포인트가 Amazon SNS 주제를 구독하게 할 수 있습니다. 이는 이 주제의 다른 예제에서 볼 수 있습니다.

다음 예제에서는 엔드포인트가 Lambda 함수입니다. 확인 토큰이 이 Lambda 함수로 전송됩니다. 수신한 후 3일 이내에 이 확인 토큰을 사용하여 구독을 확인합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

텍스트 SMS가 주제를 구독하도록 등록

동시에 여러 전화 번호로 SMS 메시지를 전송하려면 각 번호가 주제를 구독하게 합니다.

전화 번호의 구독을 시작하려면 Subscribe 작업을 사용합니다.

전달되는 파라미터에 사용되는 값에 따라 subscribe 메서드를 사용하여 서로 다른 여러 엔드포인트가 Amazon SNS 주제를 구독하게 할 수 있습니다. 이는 이 주제의 다른 예제에서 볼 수 있습니다.

다음 예제에서는 엔드포인트가 국제 통신의 표준인 E.164 형식의 전화 번호입니다.

확인 토큰이 이 전화 번호로 전송됩니다. 수신한 후 3일 이내에 이 확인 토큰을 사용하여 구독을 확인합니다.

Amazon SNS로 SMS 메시지를 전송하는 다른 방법은 AWS SDK for PHP 버전 3으로 Amazon SNS에서 SMS 메시지 전송을 참조하세요.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
```

```
}
```

주제 구독 확인

구독을 실제로 생성하려면 앞에서 설명했듯이 엔드포인트 소유자가 구독이 처음 설정되었을 때 전송된 토큰을 사용하여 주제로부터 메시지를 수신할 의도를 확인해야 합니다. 확인 토큰은 3일간 유효합니다. 3일 후 새 구독을 생성하여 토큰을 재전송할 수 있습니다.

구독을 확인하려면 ConfirmSubscription 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제 구독 나열

지정된 AWS 리전에서 최대 100개의 기존 구독을 나열하려면 ListSubscriptions 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

주제에서 구독 취소

주제를 구독하는 엔드포인트를 제거하려면 Unsubscribe 작업을 사용합니다.

구독에 삭제 인증이 필요한 경우 구독 또는 주제의 소유자만 구독 해지가 가능하고 AWS 서명이 필요합니다. unsubscribe 호출에 인증이 필요하지 않고 요청자가 구독 소유자가 아닌 경우, 최종 취소 메시지가 해당 엔드포인트로 전송됩니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Amazon SNS 주제에 메시지 게시

Amazon SNS 주제를 구독하는 각 엔드포인트에 메시지를 전송하려면 Publish 작업을 사용합니다.

메시지 텍스트 및 Amazon SNS 주제의 Amazon 리소스 이름(ARN) 등 메시지를 게시하기 위한 파라미터를 포함하는 객체를 만듭니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하여 Amazon SNS에서 SMS 메시지 보내기

사용자는 Amazon Simple Notification Service (Amazon SNS)를 사용하여 SMS 수신 가능한 디바이스에 문자 메시지 또는 SMS 메시지를 전송할 수 있습니다. 전화번호로 메시지를 직접 전송할 수 있으며, 전화번호에서 주제를 구독하고 메시지를 주제로 전송하여 메시지를 여러 전화번호로 한 번에 전송할 수 있습니다.

Amazon SNS를 사용하여 전송을 최적화하는 방법(비용 또는 안정성 있는 전송), 월 지출 한도, 메시지 전송을 로깅하는 방법, 일일 SMS 사용 보고서를 구독하는지 여부 등 SMS 메시징에 대한 기본 설정을 지정합니다. 이러한 기본 설정은 검색되고 Amazon SNS의 SMS 속성으로 설정됩니다.

SMS 메시지를 전송할 때 E.164 형식을 사용하여 전화번호를 지정합니다. E.164는 국제 통신에 사용되는 전화번호 구조의 표준입니다. 이 형식을 따르는 전화번호는 최대 15자리 숫자를 사용할 수 있으며 더하기 문자(+) 및 국가 코드가 접두사로 추가됩니다. 예를 들어, E.164 형식의 미국 전화번호는 +1001XXX5550100으로 표시될 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- GetSMSAttributes를 사용하여 계정에서 SMS 메시지 전송에 대한 기본 설정 검색
- SetSMSAttributes를 사용하여 계정에서 SMS 메시지 전송에 대한 기본 설정 업데이트
- CheckIfPhoneNumberISOptedOut을 사용하여 지정된 전화 번호 소유자가 사용자 계정에서 보내는 SMS 메시지 수신을 옵트아웃했는지 여부 확인

- ListPhoneNumberOptedOut을 사용하여 소유자가 사용자 계정에서 보내는 SMS 메시지 수신을 옵트아웃한 전화 번호 나열
- Publish를 사용하여 전화 번호로 직접 텍스트 메시지(SMS 메시지) 전송

Amazon SNS 사용에 대한 자세한 내용은 휴대폰 번호가 구독자인 경우 사용자 알림에 Amazon SNS 사용(SMS 전송)을 참조하세요.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

SMS 속성 가져오기

SMS 메시지의 기본 설정을 검색하려면 GetSMSAttributes 작업을 사용합니다.

다음 예제에서는 `DefaultSMSType` 속성을 가져옵니다. 이 속성은 SMS 메시지를 최소 비용이 발생하도록 메시지 전송 최적화하는 `Promotional`로 전송할지, 최고의 안정성을 달성하도록 메시지 전송을 최적화하는 `Transactional`로 전송할지 제어합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
```

```
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

SMS 속성 설정

SMS 메시지의 기본 설정을 업데이트하려면 SetSMSAttributes 작업을 사용합니다.

다음 예제에서는 `DefaultSMSType` 속성을 `Transactional`로 설정하여 최고의 안정성을 달성하도록 메시지 전송을 최적화합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

전화 번호가 옵트아웃되었는지 여부 확인

지정된 전화 번호 소유자가 사용자 계정에서 보내는 SMS 메시지 수신을 옵트아웃했는지 여부를 확인하려면 CheckIfPhoneNumberIsOptedOut 작업을 사용합니다.

다음 예제에서는 전화 번호가 국제 통신의 표준인 E.164 형식입니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

옵트아웃된 전화 번호 나열

소유자가 사용자 계정에서 보내는 SMS 메시지 수신을 옵트아웃한 전화 번호 목록을 검색하려면 ListPhoneNumbersOptedOut 작업을 사용합니다.

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

샘플 코드

```
$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());
}
```

텍스트 메시지(SMS 메시지)에 게시

전화 번호로 직접 텍스트 메시지(SMS 메시지)를 전송하려면 Publish 작업을 사용합니다.

다음 예제에서는 전화 번호가 국제 통신의 표준인 E.164 형식입니다.

SMS 메시지는 최대 140바이트를 포함할 수 있습니다. 단일 SMS 게시 작업에 대한 크기 제한은 1,600바이트입니다.

SMS 메시지 전송에 대한 자세한 내용은 SMS 메시지 전송을 참조하십시오.

가져옵니다.

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

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;
```

샘플 코드

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용한 Amazon SQS 예제

Amazon Simple Queue Service(SQS)는 빠르고 안정적이며 확장 가능한 완전관리형 메시지 대기열 서비스입니다. Amazon SQS를 사용하면 클라우드 애플리케이션의 구성 요소를 분리할 수 있습니다. Amazon SQS에는 높은 처리량과 최소 1회 처리 기능을 갖춘 표준 대기열과 선입선출(FIFO) 전송 및 정확히 1회 처리를 제공하는 FIFO 대기열이 포함되어 있습니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- AWS SDK for PHP 버전 3을 사용하는 Amazon SQS에서 긴 폴링 활성화
- AWS SDK for PHP 버전 3을 사용하는 Amazon SQS의 가시성 타임아웃 관리
- AWS SDK for PHP 버전 3을 사용하는 Amazon SQS에서 메시지 전송 및 수신
- AWS SDK for PHP 버전 3을 사용하여 Amazon SQS 버전 3에서 DLQ(Dead Letter Queue) 사용

- AWS SDK for PHP 버전 3을 사용하여 Amazon SQS에서 대기열 사용

AWS SDK for PHP 버전 3을 사용하는 Amazon SQS에서 긴 폴링 활성화

긴 폴링은 응답을 전송하기 전에 대기열에서 메시지를 사용할 수 있을 때까지 Amazon SQS를 지정된 시간 동안 대기시켜 빈 응답의 개수를 줄입니다. 또한, 긴 폴링은 서버의 샘플링 대신에 모든 서버를 쿼리하여 False인 빈 응답을 제거합니다. 긴 폴링을 활성화하려면 수신된 메시지에 0이 아닌 대기 시간을 지정합니다. 자세한 내용은 SQS 긴 폴링을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- SetQueueAttribute를 사용하여 Amazon SQS 대기열에 속성을 설정하여 긴 폴링이 가능하도록 합니다.
- ReceiveMessage를 사용하여 긴 폴링이 있는 하나 이상의 메시지를 검색합니다.
- CreateQueue를 사용하여 긴 폴링 대기열을 생성합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

긴 폴링을 활성화하기 위한 대기열의 속성 설정

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

긴 폴링이 있는 메시지 검색

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

긴 폴링이 있는 대기열 생성

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$queueName = "QUEUE_NAME";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하는 Amazon SQS의 가시성 타임아웃 관리

제한 시간 초과는 Amazon SQS에서 다른 사용 구성 요소가 메시지를 수신하고 처리할 수 없는 기간입니다. 자세히 알아보려면 제한 시간 초과를 참조하세요.

다음 예에서는 작업 방법을 보여줍니다.

- ChangeMessageVisibilityBatch를 사용하여 대기열에서 지정된 메시지의 제한 시간 초과를 새 값으로 변경합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

여러 메시지의 제한 시간 초과 변경

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```

$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage(array(
```

```
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 10,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
    ));
    $messages = $result->get('Messages');
    if ($messages != null) {
        $entries = array();
        for ($i = 0; $i < count($messages); $i++) {
            $entries[] = [
                'Id' => 'unique_is_msg' . $i, // REQUIRED
                'ReceiptHandle' => $messages[$i]['ReceiptHandle'], // REQUIRED
                'VisibilityTimeout' => 3600
            ];
        }
        $result = $client->changeMessageVisibilityBatch([
            'Entries' => $entries,
            'QueueUrl' => $queueUrl
        ]);

        var_dump($result);
    } else {
        echo "No messages in queue \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하는 Amazon SQS에서 메시지 전송 및 수신

Amazon SQS 메시지에 대해 자세히 알아보려면 Service Quotas 사용 설명서의 SQS 대기열로 메시지 전송 및 SQS 대기열에서 메시지 수신 및 삭제를 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- SendMessage를 사용하여 지정된 대기열에 메시지를 제공합니다.
- ReceiveMessage를 사용하여 지정된 대기열에서 하나 이상의 메시지(최대 10개)를 검색합니다.
- DeleteMessage를 사용하여 대기열에서 메시지를 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

메시지 전송

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

메시지 수신 및 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ]);
    if (!empty($result->get('Messages'))) {
        var_dump($result->get('Messages')[0]);
        $result = $client->deleteMessage([
            'QueueUrl' => $queueUrl, // REQUIRED
            'ReceiptHandle' => $result->get('Messages')[0]['ReceiptHandle'] // REQUIRED
```

```
        ]);
    } else {
        echo "No messages in queue. \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하여 Amazon SQS 버전 3에서 DLQ(Dead Letter Queue) 사용

DLQ(Dead Letter Queue)은 성공적으로 처리하지 못한 메시지를 다른 (소스) 대기열에서 보낼 수 있는 대기열입니다. DLQ(Dead Letter Queue)에서 이 메시지를 구분하고 격리하여 처리에 실패한 이유를 확인할 수 있습니다. DLQ(Dead Letter Queue)로 메시지를 보내는 각 소스 대기열을 개별적으로 구성해야 합니다. 여러 대기열에서 하나의 DLQ(Dead Letter Queue)로 메시지를 보낼 수 있습니다.

자세한 내용은 SQS DLQ(Dead Letter Queue) 사용을 참조하세요.

다음 예에서는 작업 방법을 보여줍니다.

- SetQueueAttributes를 사용하여 DLQ(Dead Letter Queue)를 활성화합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

DLQ(Dead Letter Queue) 활성화

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'RedrivePolicy' => "{\"deadLetterTargetArn\":\"DEAD_LETTER_QUEUE_ARN\",
\"maxReceiveCount\":\"10\"}"
        ],
        'QueueUrl' => $queueUrl // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하여 Amazon SQS에서 대기열 사용

Amazon SQS 대기열에 대해 자세히 알아보려면 SQS 대기열의 작동 방식을 참조하세요.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListQueues를 사용하여 대기열의 목록을 반환합니다.
- CreateQueue를 사용하여 새 대기열을 생성합니다.
- GetQueueUrl을 사용하여 기존 대기열의 URL을 반환합니다.
- DeleteQueue를 사용하여 지정된 대기열을 삭제합니다.

AWS SDK for PHP에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

보안 인증 정보

예제 코드를 실행하기 전에 the section called “를 사용하여 인증 AWS”에 설명된 대로 AWS 보안 인증을 구성합니다. 그 다음 the section called “설치”에 설명된 AWS SDK for PHP를 가져옵니다.

대기열 목록 반환

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

대기열 생성

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

대기열의 URL 반환

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$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());
}
```

대기열 삭제

가져옵니다.

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

샘플 코드

```
$queueUrl = "SQS_QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->deleteQueue([
        'QueueUrl' => $queueUrl // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

AWS SDK for PHP 버전 3을 사용하여 Amazon EventBridge 글로벌 엔드포인트로 이벤트 전송

Amazon EventBridge 글로벌 엔드포인트를 사용하여 이벤트 기반 애플리케이션의 가용성과 신뢰성을 개선할 수 있습니다.

EventBridge 글로벌 엔드포인트를 설정한 후에는 SDK for PHP를 사용하여 이벤트를 전송할 수 있습니다.

> **Important**
>
> PHP용 SDK와 함께 EventBridge 글로벌 엔드포인트를 사용하려면 PHP 환경에 AWS 공통 런타임(AWS CRT) 확장이 설치되어 있어야 합니다.

다음 예시에서는 `EventBridgeClient`의 PutEvents 메서드를 사용하여 단일 이벤트를 EventBridge 글로벌 엔드포인트로 보냅니다.

```
<?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]
]);
```

이 블로그 게시물에는 EventBridge 글로벌 엔드포인트에 대한 자세한 정보가 포함되어 있습니다.

SDK for PHP 코드 예제

이 주제의 코드 예제에서는를 AWS SDK for PHP 와 함께 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- SDK for PHP를 사용한 API Gateway 예제
- SDK for PHP를 사용한 Aurora 예제
- SDK for PHP를 사용한 Auto Scaling 예제
- SDK for PHP를 사용한 Amazon Bedrock 예시
- SDK for PHP를 사용한 Amazon Bedrock Runtime 예시
- SDK for PHP를 사용한 Amazon DocumentDB 예제
- SDK for PHP를 사용한 DynamoDB 예제
- SDK for PHP를 사용한 Amazon CE2 예제
- AWS Glue SDK for PHP를 사용한 예제
- SDK for PHP를 사용한 IAM 예제
- SDK for PHP를 사용한 Kinesis 예제
- AWS KMS SDK for PHP를 사용한 예제
- SDK for PHP를 사용한 Lambda 예제
- SDK for PHP를 사용한 Amazon MSK 예제
- SDK for PHP를 사용한 Amazon RDS 예제
- SDK for PHP를 사용한 Amazon RDS 데이터 서비스 예제
- SDK for PHP를 사용한 Amazon Rekognition 예제
- PHP SDK를 사용한 Amazon S3용 코드 예제

- SDK for PHP를 사용한 S3 디렉터리 버킷 예제
- SDK for PHP를 사용한 Amazon SES 예제
- SDK for PHP를 사용한 Amazon SNS에 대한 코드 예제
- SDK for PHP를 사용한 Amazon SQS 예제

SDK for PHP를 사용한 API Gateway 예제

다음 코드 예제에서는 API Gateway와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`GetBasePathMapping`

다음 코드 예시는 `GetBasePathMapping`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);
```

```
    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetBasePathMapping을 참조하세요.

`ListBasePathMappings`

다음 코드 예시는 `ListBasePathMappings`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */
```

```
function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListBasePathMappings를 참조하세요.

`UpdateBasePathMapping`

다음 코드 예시는 `UpdateBasePathMapping`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
```

```
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- API 세부 정보는 AWS SDK for PHP API 참조의 UpdateBasePathMapping을 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

SDK for PHP를 사용한 Aurora 예제

다음 코드 예제에서는 Aurora와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for PHP

AWS SDK for PHP 를 사용하여 Amazon RDS 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 이메일로 보내는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 이 예제에서는 RESTful PHP 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Amazon RDS 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 사용하여 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

SDK for PHP를 사용한 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling 사용을 시작하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DescribeAutoScalingGroups을 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 시작 템플릿과 가용 영역이 있는 Amazon EC2 Auto Scaling 그룹을 생성하고 실행 중인 인스턴스에 대한 정보를 가져옵니다.
- Amazon CloudWatch 지표 수집 활성화
- 그룹의 원하는 용량을 업데이트하고 인스턴스가 시작될 때까지 기다립니다.
- 그룹에서 인스턴스를 종료합니다.
- 사용자 요청 및 용량 변경에 따라 발생하는 조정 활동을 나열합니다.
- CloudWatch 지표에 대한 통계를 가져온 다음 리소스를 정리합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("---------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
```

```
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );
```

```
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
```

```
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
```

```
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
```

```
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
```

```
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하십시오.
  - CreateAutoScalingGroup

- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

작업

`CreateAutoScalingGroup`

다음 코드 예시는 `CreateAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
```

```
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- API에 대한 자세한 설명은 AWS SDK for PHP API 참조 문서의 CreateAutoScalingGroup을 참조하세요.

`DeleteAutoScalingGroup`

다음 코드 예시는 `DeleteAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- API에 대한 자세한 설명은 AWS SDK for PHP API 참조 문서의 DeleteAutoScalingGroup을 참조하세요.

`DescribeAutoScalingGroups`

다음 코드 예시는 `DescribeAutoScalingGroups`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DescribeAutoScalingGroups를 참조하세요.

`DescribeAutoScalingInstances`

다음 코드 예시는 `DescribeAutoScalingInstances`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DescribeAutoScalingInstances를 참조하세요.

`DescribeScalingActivities`

다음 코드 예시는 `DescribeScalingActivities`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DescribeScalingActivities를 참조하세요.

`DisableMetricsCollection`

다음 코드 예시는 `DisableMetricsCollection`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function disableMetricsCollection($autoScalingGroupName)
```

```
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DisableMetricsCollection을 참조하세요.

`EnableMetricsCollection`

다음 코드 예시는 `EnableMetricsCollection`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 EnableMetricsCollection을 참조하세요.

`SetDesiredCapacity`

다음 코드 예시는 `SetDesiredCapacity`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'DesiredCapacity' => $desiredCapacity,
    ]);
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 SetDesiredCapacity를 참조하세요.

`TerminateInstanceInAutoScalingGroup`

다음 코드 예시는 `TerminateInstanceInAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 TerminateInstanceInAutoScalingGroup을 참조하세요.

`UpdateAutoScalingGroup`

다음 코드 예시는 `UpdateAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
    }
```

- API에 대한 자세한 설명은 AWS SDK for PHP API 참조 문서의 UpdateAutoScalingGroup을 참조하세요.

SDK for PHP를 사용한 Amazon Bedrock 예시

다음 코드 예제에서는 AWS SDK for PHP Amazon Bedrock에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`ListFoundationModels`

다음 코드 예시는 `ListFoundationModels`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

사용 가능한 Amazon Bedrock 기초 모델을 나열하세요.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListFoundationModels를 참조하세요.

SDK for PHP를 사용한 Amazon Bedrock Runtime 예시

다음 코드 예제에서는 Amazon Bedrock 런타임과 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오
- Amazon Nova
- Amazon Titan Image Generator

- Anthropic Claude
- Stable Diffusion

시나리오

Amazon Bedrock에서 여러 파운데이션 모델 간접 호출

다음 코드 예제에서는 Amazon Bedrock에서 프롬프트를 준비하고 다양한 대규모 언어 모델(LLM)에 전송하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Bedrock에서 여러 LLM을 간접 호출합니다.

```
namespace BedrockRuntime;

class GettingStartedWithBedrockRuntime
{
    protected BedrockRuntimeService $bedrockRuntimeService;
    public function runExample()
    {
        echo "\n";
        echo "---------------------------------------------------------------------
\n";
        echo "Welcome to the Amazon Bedrock Runtime getting started demo using PHP!
\n";
        echo "---------------------------------------------------------------------
\n";
        $bedrockRuntimeService = new BedrockRuntimeService();
        $prompt = 'In one paragraph, who are you?';
        echo "\nPrompt: " . $prompt;
        echo "\n\nAnthropic Claude:\n";
        echo $bedrockRuntimeService->invokeClaude($prompt);
        echo
 "\n---------------------------------------------------------------------\n";
```

```
        $image_prompt = 'stylized picture of a cute old steampunk robot';
        echo "\nImage prompt: " . $image_prompt;
        echo "\n\nStability.ai Stable Diffusion XL:\n";
        $diffusionSeed = rand(0, 4294967295);
        $style_preset = 'photographic';
        $base64 = $bedrockRuntimeService->invokeStableDiffusion($image_prompt,
 $diffusionSeed, $style_preset);
        $image_path = $this->saveImage($base64, 'stability.stable-diffusion-xl');
        echo "The generated image has been saved to $image_path";
        echo "\n\nAmazon Titan Image Generation:\n";
        $titanSeed = rand(0, 2147483647);
        $base64 = $bedrockRuntimeService->invokeTitanImage($image_prompt,
 $titanSeed);
        $image_path = $this->saveImage($base64, 'amazon.titan-image-generator-v2');
        echo "The generated image has been saved to $image_path";
    }

    private function saveImage($base64_image_data, $model_id): string
    {
        $output_dir = "output";
        if (!file_exists($output_dir)) {
            mkdir($output_dir);
        }

        $i = 1;
        while (file_exists("$output_dir/$model_id" . '_' . "$i.png")) {
            $i++;
        }

        $image_data = base64_decode($base64_image_data);
        $file_path = "$output_dir/$model_id" . '_' . "$i.png";
        $file = fopen($file_path, 'wb');
        fwrite($file, $image_data);
        fclose($file);
        return $file_path;
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하세요.
  - InvokeModel
  - InvokeModelWithResponseStream

Amazon Nova

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Amazon Nova로 텍스트 메시지를 전송하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Amazon Nova로 텍스트 메시지를 전송합니다.

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

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

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

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

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

```
                "content" => [["text" => $userMessage]]
            ]
        ];

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

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

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

- API 세부 정보는 AWS SDK for PHP API 참조의 Converse를 참조하세요.

Amazon Titan Image Generator

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Amazon Titan Image를 간접 호출하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Titan Image Generator를 사용하여 이미지를 생성합니다.

```
    public function invokeTitanImage(string $prompt, int $seed)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and default values for Titan Image models refer
 to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html

        $base64_image_data = "";
        try {
            $modelId = 'amazon.titan-image-generator-v2:0';
            $request = json_encode([
                'taskType' => 'TEXT_IMAGE',
                'textToImageParams' => [
                    'text' => $prompt
                ],
                'imageGenerationConfig' => [
                    'numberOfImages' => 1,
                    'quality' => 'standard',
                    'cfgScale' => 8.0,
                    'height' => 512,
                    'width' => 512,
                    'seed' => $seed
                ]
            ]);
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => $request,
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
```

```
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 InvokeModel을 참조하세요.

Anthropic Claude

InvokeModel

다음 코드 예제에서는 Invoke Model API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Anthropic Claude 2 파운데이션 모델을 간접 호출하여 텍스트를 생성합니다.

```
    public function invokeClaude($prompt)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and default values for Anthropic Claude, refer
 to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
claude.html

        $completion = "";
        try {
            $modelId = 'anthropic.claude-3-haiku-20240307-v1:0';
        // Claude requires you to enclose the prompt as follows:
            $body = [
```

```
                'anthropic_version' => 'bedrock-2023-05-31',
                'max_tokens' => 512,
                'temperature' => 0.5,
                'messages' => [[
                    'role' => 'user',
                    'content' => $prompt
                ]]
            ];
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $completion = $response_body->content[0]->text;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $completion;
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 InvokeModel을 참조하세요.

Stable Diffusion

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Stability.ai Stable Diffusion XL 모델을 간접적으로 호출하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Stable Diffusion을 사용하여 이미지를 생성합니다.

```
    public function invokeStableDiffusion(string $prompt, int $seed, string
 $style_preset)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available style_presets of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-diffusion-xl-v1';
            $body = [
                'text_prompts' => [
                    ['text' => $prompt]
                ],
                'seed' => $seed,
                'cfg_scale' => 10,
                'steps' => 30
            ];
            if ($style_preset) {
                $body['style_preset'] = $style_preset;
            }

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->artifacts[0]->base64;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 InvokeModel을 참조하세요.

SDK for PHP를 사용한 Amazon DocumentDB 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon DocumentDB에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

서버리스 예제

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
```

```
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

SDK for PHP를 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 영화 데이터를 저장할 수 있는 테이블을 생성합니다.
- 테이블에 하나의 영화를 추가하고 가져오고 업데이트합니다.
- 샘플 JSON 파일에서 테이블에 영화 데이터를 씁니다.
- 특정 연도에 개봉된 영화를 쿼리합니다.
- 특정 연도 범위 동안 개봉된 영화를 스캔합니다.
- 테이블에서 영화를 삭제한 다음, 테이블을 삭제합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제에서는 지원 파일을 사용하므로 PHP 예제 README.md 파일의 지침을 읽어야 합니다.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
```

```
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
```

```
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";
```

```
        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
```

```
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하세요.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

작업

`BatchExecuteStatement`

다음 코드 예시는 `BatchExecuteStatement`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
```

```
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 BatchExecuteStatement를 참조하세요.

`BatchWriteItem`

다음 코드 예시는 `BatchWriteItem`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public function writeBatch(string $TableName, array $Batch, int $depth = 2)
   {
       if (--$depth <= 0) {
           throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
       }

       $marshal = new Marshaler();
       $total = 0;
       foreach (array_chunk($Batch, 25) as $Items) {
           foreach ($Items as $Item) {
               $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
           }
           try {
               echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
               $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
               $BatchWrite = [];
           } catch (Exception $e) {
               echo "uh oh...";
               echo $e->getMessage();
               die();
           }
           if ($total >= 250) {
               echo "250 movies is probably enough. Right? We can stop there.\n";
               break;
           }
       }
   }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 BatchWriteItem을 참조하세요.

`CreateTable`

다음 코드 예시는 `CreateTable`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블을 생성합니다.

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateTable을 참조하세요.

`DeleteItem`

다음 코드 예시는 `DeleteItem`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteItem을 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteTable을 참조하세요.

`ExecuteStatement`

다음 코드 예시는 `ExecuteStatement`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
```

```
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ExecuteStatement를 참조하세요.

`GetItem`

다음 코드 예시는 `GetItem`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetItem을 참조하세요.

`ListTables`

다음 코드 예시는 `ListTables`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
            'ExclusiveStartTableName' => $exclusiveStartTableName,
```

```
            'Limit' => $limit,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListTables를 참조하세요.

`PutItem`

다음 코드 예시는 `PutItem`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
```

```
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 PutItem을 참조하세요.

`Query`

다음 코드 예시는 `Query`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);

    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
```

```
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Query를 참조하세요.

Scan

다음 코드 예시는 Scan의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
```

```
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Scan을 참조하세요.

`UpdateItem`

다음 코드 예시는 `UpdateItem`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
```

```
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 UpdateItem을 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway

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

PartiQL 문 배치를 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 여러 SELECT 문을 실행하여 항목 배치를 가져옵니다.
- 여러 INSERT 문을 실행하여 항목 배치를 추가합니다.
- 여러 UPDATE 문을 실행하여 항목 배치를 업데이트합니다.
- 여러 DELETE 문을 실행하여 항목 배치를 삭제합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("---------------------------------------\n");
```

```
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("---------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);
```

```
        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
```

```
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
```

```
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
```

```
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 BatchExecuteStatement를 참조하세요.

PartiQL을 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- SELECT 문을 실행하여 항목을 가져옵니다.
- INSERT 문을 실행하여 항목을 추가합니다.
- UPDATE 문을 실행하여 항목을 업데이트합니다.
- DELETE 문을 실행하여 항목을 삭제합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
```

```
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
```

```
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
```

```
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
```

```
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ExecuteStatement를 참조하세요.

서버리스 예제

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data
```

```
            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

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

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
```

```
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
```

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

SDK for PHP를 사용한 Amazon CE2 예제

다음 코드 예제에서는 Amazon EC2와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateVpc`

다음 코드 예시는 `CreateVpc`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
```

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

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateVpc를 참조하세요.

`CreateVpcEndpoint`

다음 코드 예시는 `CreateVpcEndpoint`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

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

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateVpcEndpoint를 참조하세요.

`DeleteVpc`

다음 코드 예시는 `DeleteVpc`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteVpc를 참조하세요.

`DeleteVpcEndpoints`

다음 코드 예시는 `DeleteVpcEndpoints`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteVpcEndpoints를 참조하세요.

`DescribeRouteTables`

다음 코드 예시는 DescribeRouteTables의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DescribeRouteTables을 참조하세요.

AWS Glue SDK for PHP를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for PHP 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Glue.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 퍼블릭 Amazon S3 버킷을 크롤링하고 CSV 형식의 메타데이터 데이터베이스를 생성하는 크롤러를 생성합니다.
- 의 데이터베이스 및 테이블에 대한 정보를 나열합니다 AWS Glue Data Catalog.
- 작업을 생성하여 S3 버킷에서 CSV 데이터를 추출하고, 데이터를 변환하며, JSON 형식의 출력을 다른 S3 버킷으로 로드합니다.
- 작업 실행에 대한 정보를 나열하고 변환된 데이터를 확인하며 리소스를 정리합니다.

자세한 내용은 자습서: AWS Glue Studio 시작하기를 참조하세요.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");
```

```
        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
```

```
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
```

```
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
```

```
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

```
    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
```

```
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
```

```
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

작업

`CreateCrawler`

다음 코드 예시는 `CreateCrawler`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateCrawler를 참조하세요.

CreateJob

다음 코드 예시는 CreateJob의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateJob을 참조하세요.

DeleteCrawler

다음 코드 예시는 DeleteCrawler의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteCrawler를 참조하세요.

`DeleteDatabase`

다음 코드 예시는 `DeleteDatabase`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
```

```
    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteDatabase를 참조하세요.

`DeleteJob`

다음 코드 예시는 `DeleteJob`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteJob을 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteTable을 참조하세요.

`GetCrawler`

다음 코드 예시는 `GetCrawler`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
```

```
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetCrawler를 참조하세요.

`GetDatabase`

다음 코드 예시는 `GetDatabase`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetDatabase를 참조하세요.

`GetJobRun`

다음 코드 예시는 `GetJobRun`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetJobRun을 참조하세요.

`GetJobRuns`

다음 코드 예시는 `GetJobRuns`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetJobRuns를 참조하세요.

`GetTables`

다음 코드 예시는 `GetTables`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    $databaseName = "doc-example-database-$uniqid";

    $tables = $glueService->getTables($databaseName);

public function getTables($databaseName): Result
{
    return $this->glueClient->getTables([
        'DatabaseName' => $databaseName,
    ]);
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetTables를 참조하세요.

`ListJobs`

다음 코드 예시는 `ListJobs`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    $jobs = $glueService->listJobs();
    echo "Current jobs:\n";
    foreach ($jobs['JobNames'] as $jobsName) {
        echo "{$jobsName}\n";
    }
```

```
    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListJobs를 참조하세요.

`StartCrawler`

다음 코드 예시는 `StartCrawler`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
```

```
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 StartCrawler를 참조하세요.

`StartJobRun`

다음 코드 예시는 `StartJobRun`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 StartJobRun을 참조하세요.

SDK for PHP를 사용한 IAM 예제

다음 코드 예제에서는 IAM과 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.

- 역할을 수임하고 임시 자격 증명 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
```

```
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
```

```
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

작업

`AttachRolePolicy`

다음 코드 예시는 `AttachRolePolicy`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";
```

```
$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 AttachRolePolicy를 참조하세요.

`CreatePolicy`

다음 코드 예시는 `CreatePolicy`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";
```

```
    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
{
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreatePolicy를 참조하세요.

`CreateRole`

다음 코드 예시는 `CreateRole`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
```

```
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateRole을 참조하세요.

`CreateServiceLinkedRole`

다음 코드 예시는 `CreateServiceLinkedRole`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
```

```
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateServiceLinkedRole을 참조하세요.

`CreateUser`

다음 코드 예시는 `CreateUser`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
```

```
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateUser를 참조하세요.

`GetAccountPasswordPolicy`

다음 코드 예시는 `GetAccountPasswordPolicy`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetAccountPasswordPolicy를 참조하세요.

`GetPolicy`

다음 코드 예시는 `GetPolicy`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetPolicy를 참조하세요.

`GetRole`

다음 코드 예시는 `GetRole`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
```

```
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetRole을 참조하세요.

`ListAttachedRolePolicies`

다음 코드 예시는 `ListAttachedRolePolicies`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListAttachedRolePolicies를 참조하세요.

ListGroups

다음 코드 예시는 ListGroups의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListGroups를 참조하세요.

ListPolicies

다음 코드 예시는 ListPolicies의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListPolicies를 참조하세요.

`ListRolePolicies`

다음 코드 예시는 `ListRolePolicies`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListRolePolicies를 참조하세요.

`ListRoles`

다음 코드 예시는 `ListRoles`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
```

```
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListRoles를 참조하세요.

`ListSAMLProviders`

다음 코드 예시는 `ListSAMLProviders`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListSAMLProviders를 참조하세요.

`ListUsers`

다음 코드 예시는 `ListUsers`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListUsers를 참조하세요.

SDK for PHP를 사용한 Kinesis 예제

다음 코드 예제에서는 Kinesis와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

서버리스 예제

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
```

```
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

AWS KMS SDK for PHP를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for PHP 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS KMS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

안녕하세요 AWS KMS

다음 코드 예제에서는 AWS Key Management Service를 사용하여 시작하는 방법을 보여 줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListKeys를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- KMS 키를 생성합니다.
- 계정의 KMS 키를 나열하고 해당 키에 대한 세부 정보를 확인하세요.
- KMS 키를 활성화 및 비활성화합니다.
- 클라이언트 측 암호화에 사용할 수 있는 대칭 데이터 키를 생성하세요.
- 데이터에 디지털 방식으로 서명하는 데 사용되는 비대칭 키를 생성합니다.
- 키에 태그를 지정합니다.
- KMS 키를 삭제합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

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

This KMS Basics scenario creates two key types:
```

```
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

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

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

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

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

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

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

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

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

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
```

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

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

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

```
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

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

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

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
```

```
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

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

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

```
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

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

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
changes to PendingDeletion and the key can't be used in any cryptographic
operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

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

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

```
            $this->cleanUp();
        }

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



namespace Kms;

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

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

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


    /***
     * @param string $keySpec
```

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



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
"SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
```

```
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
```

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



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

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



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



    /***
     * @return array
```

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



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



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

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


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



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
```

```
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
```

```
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
```

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



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



    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
```

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


}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하세요.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - 암호화
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

작업

`CreateAlias`

다음 코드 예시는 `CreateAlias`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateAlias를 참조하세요.

`CreateGrant`

다음 코드 예시는 `CreateGrant`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /***
    * @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;
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateGrant를 참조하세요.

`CreateKey`

다음 코드 예시는 `CreateKey`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /***
    * @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;
```

```
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateKey를 참조하세요.

`Decrypt`

다음 코드 예시는 `Decrypt`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 Decrypt를 참조하세요.

`DeleteAlias`

다음 코드 예시는 `DeleteAlias`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DeleteAlias를 참조하세요.

`DescribeKey`

다음 코드 예시는 `DescribeKey`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DescribeKey를 참조하세요.

`DisableKey`

다음 코드 예시는 `DisableKey`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DisableKey를 참조하세요.

`EnableKey`

다음 코드 예시는 `EnableKey`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 EnableKey를 참조하세요.

`Encrypt`

다음 코드 예시는 `Encrypt`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 Encrypt를 참조하세요.

`ListAliases`

다음 코드 예시는 `ListAliases`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListAliases를 참조하세요.

`ListGrants`

다음 코드 예시는 `ListGrants`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListGrants를 참조하세요.

`ListKeys`

다음 코드 예시는 `ListKeys`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @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;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListKeys를 참조하세요.

PutKeyPolicy

다음 코드 예시는 PutKeyPolicy의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 PutKeyPolicy를 참조하세요.

`RevokeGrant`

다음 코드 예시는 `RevokeGrant`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 RevokeGrant를 참조하세요.

ScheduleKeyDeletion

다음 코드 예시는 ScheduleKeyDeletion의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 ScheduleKeyDeletion을 참조하세요.

Sign

다음 코드 예시는 Sign의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 Sign을 참조하세요.

`TagResource`

다음 코드 예시는 `TagResource`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 TagResource를 참조하세요.

SDK for PHP를 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할과 Lambda 함수를 생성하고 핸들러 코드를 업로드합니다.
- 단일 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다.
- 함수 코드를 업데이트하고 환경 변수로 구성합니다.
- 새 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다. 반환된 실행 로그를 표시합니다.
- 계정의 함수를 나열합니다.

자세한 내용은 콘솔로 Lambda 함수 생성을 참조하세요.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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";
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - 간접 호출
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

작업

`CreateFunction`

다음 코드 예시는 `CreateFunction`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
```

```
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateFunction을 참조하세요.

`DeleteFunction`

다음 코드 예시는 `DeleteFunction`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteFunction을 참조하세요.

`GetFunction`

다음 코드 예시는 `GetFunction`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetFunction을 참조하세요.

`Invoke`

다음 코드 예시는 `Invoke`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
```

```
            'LogType' => $logType,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 간접 호출을 참조하세요.

`ListFunctions`

다음 코드 예시는 `ListFunctions`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function listFunctions($maxItems = 50, $marker = null)
    {
        if (is_null($marker)) {
            return $this->lambdaClient->listFunctions([
                'MaxItems' => $maxItems,
            ]);
        }

        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListFunctions를 참조하세요.

`UpdateFunctionCode`

다음 코드 예시는 `UpdateFunctionCode`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 UpdateFunctionCode를 참조하세요.

`UpdateFunctionConfiguration`

다음 코드 예시는 `UpdateFunctionConfiguration`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
```

```
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 UpdateFunctionConfiguration을 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
```

```
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
```

```
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

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

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
```

```
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
```

```
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

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

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
```

```
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

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

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

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

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

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

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 S3 이벤트 사용.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

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

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function
```

```
Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}
```

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

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
```

```
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

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

SDK for PHP를 사용한 Amazon MSK 예제

다음 코드 예제에서는 Amazon MSK와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 서버리스 예제

서버리스 예제

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

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

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

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

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

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

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

SDK for PHP를 사용한 Amazon RDS 예제

다음 코드 예제에서는 Amazon RDS와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

- 서버리스 예제

작업

`CreateDBInstance`

다음 코드 예시는 `CreateDBInstance`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
```

```
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateDBInstance를 참조하세요.

`CreateDBSnapshot`

다음 코드 예시는 `CreateDBSnapshot`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
```

```
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateDBSnapshot을 참조하세요.

`DeleteDBInstance`

다음 코드 예시는 `DeleteDBInstance`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteDBInstance를 참조하세요.

`DescribeDBInstances`

다음 코드 예시는 `DescribeDBInstances`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
```

```
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DescribeDBInstances를 참조하세요.

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for PHP

AWS SDK for PHP 를 사용하여 Amazon RDS 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 이메일로 보내는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 이 예제에서는 RESTful PHP 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Amazon RDS 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 사용하여 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
```

```
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
```

```
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

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

SDK for PHP를 사용한 Amazon RDS 데이터 서비스 예제

다음 코드 예제에서는 Amazon RDS Data Service와 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for PHP

AWS SDK for PHP 를 사용하여 Amazon RDS 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 이메일로 보내는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 이 예제에서는 RESTful PHP 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Amazon RDS 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 사용하여 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

SDK for PHP를 사용한 Amazon Rekognition 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon Rekognition에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

PHP SDK를 사용한 Amazon S3용 코드 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon S3에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon S3

다음 코드 예제에서는 Amazon S3 사용을 시작하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
use Aws\S3\S3Client;

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

- API 세부 정보는 AWS SDK for PHP API 참조의 ListBuckets를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo("\n");
        echo("-------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("-------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
```

```
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
```

```
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

작업

`CopyObject`

다음 코드 예시는 `CopyObject`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

간단하게 객체를 복사합니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
```

```
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CopyObject를 참조하세요.

`CreateBucket`

다음 코드 예시는 `CreateBucket`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 만듭니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
```

```
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateBucket을 참조하세요.

`DeleteBucket`

다음 코드 예시는 `DeleteBucket`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

빈 버킷을 삭제합니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteBucket을 참조하세요.

`DeleteObject`

다음 코드 예시는 `DeleteObject`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteObject를 참조하세요.

`DeleteObjects`

다음 코드 예시는 `DeleteObjects`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

키 목록에서 객체 세트를 삭제합니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteObjects를 참조하세요.

GetObject

다음 코드 예시는 GetObject의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 가져옵니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetObject를 참조하세요.

ListObjectsV2

다음 코드 예시는 ListObjectsV2의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체를 나열합니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListObjectsV2를 참조하세요.

`PutObject`

다음 코드 예시는 `PutObject`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷에 객체를 업로드합니다.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 PutObject를 참조하세요.

시나리오

미리 서명된 URL 생성

다음 코드 예제는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
        }
    }

}
```

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

서버리스 예제

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 S3 이벤트 사용.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
```

```
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

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

SDK for PHP를 사용한 S3 디렉터리 버킷 예제

다음 코드 예제에서는 S3 디렉터리 버킷과 AWS SDK for PHP 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 기본 사항

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- VPC 및 VPC 엔드포인트를 설정합니다.
- S3 디렉터리 버킷 및 S3 Express One Zone 스토리지 클래스로 작업하도록 정책, 역할 및 사용자를 설정합니다.
- 두 개의 S3 클라이언트를 만듭니다.
- 두 개의 버킷 만들기
- 객체를 만들고 복사합니다.
- 성능 차이를 보여줍니다.
- 버킷을 채워 사전식 순서 차이를 표시합니다.
- 사용자에게 리소스를 정리할지 묻는 프롬프트를 표시합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 디렉터리 버킷 및 S3 Express One Zone의 기본 사항을 보여주는 시나리오를 실행합니다.

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

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

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

        $uuid = uniqid();

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

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

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
```

```
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

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

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

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

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

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

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

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

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

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it to
the normal bucket.\n";
```

```
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

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

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

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

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

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
```

```
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

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

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
 normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
 manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
```

```
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
 while the directory bucket does not. This is because the normal bucket considers
 the whole \"key\" to be the object identifies, while the directory bucket actually
 creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;
```

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

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

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

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

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

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

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

```
    }


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



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

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



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

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



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

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

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

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



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

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



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



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

```
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



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

```
    }



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

}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

SDK for PHP를 사용한 Amazon SES 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon SES에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for PHP

AWS SDK for PHP 를 사용하여 Amazon RDS 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 이메일로 보내는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 이 예제에서는 RESTful PHP 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Amazon RDS 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 사용하여 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

SDK for PHP를 사용한 Amazon SNS에 대한 코드 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon SNS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CheckIfPhoneNumberIsOptedOut`

다음 코드 예시는 `CheckIfPhoneNumberIsOptedOut`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 CheckIfPhoneNumberIsOptedOut을 참조하세요.

`ConfirmSubscription`

다음 코드 예시는 `ConfirmSubscription`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ConfirmSubscription을 참조하세요.

`CreateTopic`

다음 코드 예시는 `CreateTopic`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 CreateTopic을 참조하세요.

`DeleteTopic`

다음 코드 예시는 `DeleteTopic`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteTopic을 참조하세요.

`GetSMSAttributes`

다음 코드 예시는 `GetSMSAttributes`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 GetSMSAttributes를 참조하세요.

`GetTopicAttributes`

다음 코드 예시는 `GetTopicAttributes`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetTopicAttributes를 참조하세요.

`ListPhoneNumbersOptedOut`

다음 코드 예시는 `ListPhoneNumbersOptedOut`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 ListPhoneNumbersOptedOut을 참조하세요.

`ListSubscriptions`

다음 코드 예시는 `ListSubscriptions`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
```

```
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListSubscriptions를 참조하세요.

`ListTopics`

다음 코드 예시는 `ListTopics`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListTopics를 참조하세요.

`Publish`

다음 코드 예시는 `Publish`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하세요.

`SetSMSAttributes`

다음 코드 예시는 `SetSMSAttributes`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 SetSMSAttributes를 참조하세요.

`SetTopicAttributes`

다음 코드 예시는 `SetTopicAttributes`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 SetTopicAttributes를 참조하세요.

`Subscribe`

다음 코드 예시는 `Subscribe`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 주소로 주제 구독.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

HTTP 엔드포인트에서 주제를 구독합니다.

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Subscribe를 참조하세요.

`Unsubscribe`

다음 코드 예시는 `Unsubscribe`의 사용 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.

- API 세부 정보는 AWS SDK for PHP API 참조의 Unsubscribe를 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

SMS 문자 메시지 게시

다음 코드 예제에서는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하세요.

서버리스 예제

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
```

```
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

SDK for PHP를 사용한 Amazon SQS 예제

다음 코드 예제에서는 AWS SDK for PHP Amazon SQS를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 서버리스 예제

서버리스 예제

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}
```

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

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// 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);
```

AWS SDK for PHP 버전 2에서 마이그레이션

이 주제에서는 AWS SDK for PHP 버전 3을 사용하도록 코드를 마이그레이션하는 방법과 새 버전이 SDK 버전 2와 어떻게 다른지를 보여줍니다.

> **Note**
>
> SDK의 기본 사용 패턴(`$result = $client->operation($params);`)은 버전 2와 버전 3에서 달라지지 않았으므로, 마이그레이션이 원활히 진행됩니다.

소개

AWS SDK for PHP 버전 3에서는 SDK의 기능을 개선하고, 2년 동안 수집된 고객 피드백을 통합하고, 종속 항목을 업그레이드하고, 성능을 개선하며, 최신 PHP 표준을 채택하는 데 많은 노력을 기울였습니다.

버전 3의 새로운 기능

AWS SDK for PHP 버전 3에서는 PSR-4 및 PSR-7 표준을 따르며, 앞으로는 SemVer 표준을 준수할 예정입니다.

기타 새로운 기능은 다음과 같습니다.

- 서비스 클라이언트 동작을 사용자 지정하기 위한 미들웨어 시스템
- 페이지 지정된 결과를 반복하는 유연한 페이지네이터
- JMESPath를 사용하여 결과 및 페이징 개체의 데이터를 쿼리하는 기능
- `'debug'` 구성 옵션을 통해 손쉽게 디버깅

분리된 HTTP 계층

- 기본적으로 Guzzle 6가 요청을 전송하는 데 사용되지만, Guzzle 5도 지원됩니다.
- SDK는 cURL을 사용할 수 없는 환경에서 작동합니다.
- 사용자 지정 HTTP 핸들러도 지원됩니다.

비동기 요청

- 또한 waiter, 멀티파트 업로더 등과 같은 기능을 비동기로 사용할 수 있습니다.
- promise 및 코루틴을 사용하여 비동기 워크플로를 생성할 수 있습니다.
- 동시 또는 배치 요청의 성능이 향상되었습니다.

버전 2와의 차이점

프로젝트 종속 항목이 업데이트됨

SDK의 종속 항목이 이 버전에서 변경되었습니다.

- 이제 SDK를 사용하려면 PHP 8.1 이상이 필요합니다. SDK 코드 내에서 생성기를 자유롭게 사용합니다.
- SDK에서 AWS 서비스에 요청을 전송하는 데 사용되는 기본 HTTP 클라이언트 구현을 제공하는 Guzzle 6(또는 5)를 사용하도록 SDK를 업그레이드했습니다. 최신 버전의 Guzzle에서는 비동기 요청, 스왑 가능한 HTTP 핸들러, PSR-7 규정 준수, 성능 등 다양한 개선이 이루어졌습니다.
- PHP-FIG(`psr/http-message`)의 PSR-7 패키지에서는 HTTP 요청, HTTP 응답, URL, 스트림 등을 나타내는 인터페이스를 정의합니다. 이러한 인터페이스는 SDK 및 Guzzle 전반에서 사용되므로 다른 PSR-7 규격 패키지와 상호 호환됩니다.
- Guzzle의 PSR-7 구현(`guzzlehttp/psr7`)은 PSR-7의 인터페이스 구현과 여러 유용한 클래스 및 함수를 제공합니다. SDK와 Guzzle 6는 모두 이 패키지에 주로 의존합니다.
- Guzzle의 Promises/A+ 구현(`guzzlehttp/promises`)은 SDK 및 Guzzle 전반에서 비동기 요청 및 코루틴 관리 인터페이스를 제공하는 데 사용됩니다. Guzzle의 다중 cURL HTTP 핸들러는 비동기 요청에 허용되는 비차단형 I/O 모델을 궁극적으로 구현하지만, 이 패키지는 해당 패러다임 내에서 프로그래밍할 수 있습니다. 자세한 내용은 PHP용 AWS SDK for PHP버전 3의 약속을 참조하세요.
- JMESPath의 PHP 구현(`mtdowling/jmespath.php`)은 SDK에서 `Aws\Result::search()` 및 `Aws\ResultPaginator::search()` 메서드의 데이터 쿼리 기능을 제공하는 데 사용됩니다. 자세한 내용은 AWS SDK for PHP 버전 3의 JMESPath 표현식을 참조하세요.

리전 및 버전 옵션이 이제 필수임

서비스에 대한 클라이언트를 인스턴스화할 때 `'region'` 및 `'version'` 옵션을 지정합니다. AWS SDK for PHP 버전 2에서 `'version'`은 완전히 선택 사항이고, `'region'`은 경우에 따라 선택 사항

입니다. 버전 3에서는 둘 모두 항상 필요합니다. 두 옵션을 명시적으로 지정하면 코딩 중인 API 버전 및 AWS 리전을 잠글 수 있습니다. 새 API 버전이 생성되거나 새 AWS 리전이 사용 가능해질 경우 구성을 명시적으로 업데이트할 준비가 될 때까지 변경할 수 없도록 격리됩니다.

> **Note**
>
> 사용하는 API 버전이 상관없는 경우 `'version'` 옵션을 `'latest'`로 설정할 수 있습니다. 하지만 프로덕션 코드에 대해 API 버전 번호를 명시적으로 설정하는 것이 좋습니다.
> 모든 서비스를 모든 AWS 리전에서 사용할 수 있는 것은 아닙니다. 이전 및 엔드포인트 참조를 사용하여 사용 가능한 리전 목록을 확인할 수 있습니다.
> 전역적 엔드포인트 하나(예: Amazon Route 53, AWS Identity and Access Management, Amazon CloudFront)를 통해서만 서비스를 사용할 수 있는 경우, 구성된 리전을 `us-east-1`로 설정하여 클라이언트를 인스턴스화합니다.

> **Important**
>
> 또한 SDK에는 명령 파라미터로 제공된 파라미터(`@region`)에 따라 다른 AWS 리전으로 요청을 디스패치할 수 있는 다중 리전 클라이언트가 포함되어 있습니다. 이러한 클라이언트에서 기본적으로 사용되는 리전은 클라이언트 생성자에 제공된 `region` 옵션으로 지정됩니다.

클라이언트 인스턴스화 시 생성자 사용

AWS SDK for PHP 버전 3에서는 클라이언트를 인스턴스화하는 방법이 달라졌습니다. 버전 2의 `factory` 메서드 대신 `new` 키워드를 사용하여 클라이언트를 간단히 인스턴스화할 수 있습니다.

```
use Aws\DynamoDb\DynamoDbClient;

// Version 2 style
$client = DynamoDbClient::factory([
    'region'  => 'us-east-2'
]);

// Version 3 style
$client = new DynamoDbClient([
    'region'  => 'us-east-2',
    'version' => '2012-08-10'
]);
```

> **Note**
>
> `factory()` 메서드를 통한 클라이언트 인스턴스화도 계속 작동합니다. 하지만 사용되지 않는 것으로 간주됩니다.

클라이언트 구성이 변경됨

AWS SDK for PHP 버전 3의 클라이언트 구성 옵션이 버전 2와 약간 다르게 변경되었습니다. 지원되는 모든 옵션에 대한 설명은 PHP용 AWS SDK for PHP 버전 3의 구성 페이지를 참조하세요.

> **Important**
>
> 버전 3에서는 `'key'` 및 `'secret'`이 더 이상 루트 수준에서 유효한 옵션이 아니지만, `'credentials'` 옵션의 일부로 전달할 수 있습니다. 개발자가 AWS 보안 인증을 프로젝트에 하드 코딩하지 않는 것이 좋기 때문입니다.

SDK 객체

AWS SDK for PHP 버전 3에서는 `Aws\Sdk`를 대신하기 위해 `Aws\Common\Aws` 객체를 도입했습니다. `Sdk` 객체는 클라이언트 팩토리 역할을 하며 여러 클라이언트 간의 공유 구성 옵션을 관리하는 데 사용됩니다.

SDK 버전 2의 `Aws` 클래스는 서비스 로케이터처럼 작동하지만(항상 동일한 클라이언트 인스턴스 반환) 버전 3의 `Sdk` 클래스는 사용할 때마다 새로운 클라이언트 인스턴스를 반환합니다.

또한 `Sdk` 객체는 SDK 버전 2와 동일한 구성 파일 형식을 지원하지 않습니다. 이 구성 형식은 Guzzle 3에 특정하며 이제 사용되지 않습니다. 구성은 기본 배열을 사용하여 간단히 수행할 수 있으며 SDK 클래스 사용에 설명되어 있습니다.

일부 API 결과가 변경됨

SDK가 API 작업 결과를 구문 분석하는 방법에 일관성을 기하기 위해, Amazon ElastiCache, Amazon RDS, 및 Amazon Redshift는 이제 일부 API 응답에 래핑 요소를 추가했습니다.

예를 들어 버전 3에서 Amazon RDS DescribeEngineDefaultParameters 결과를 호출하면 이제 “EngineDefaults” 래핑 요소가 포함됩니다. 버전 2에는 이 요소가 없습니다.

```
$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'];
```

이 변경의 영향을 받고 결과 출력에 래핑 요소가 포함되는 작업은 다음과 같습니다(아래 괄호 안에 제공됨).

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress(CacheSecurityGroup)
  - CopySnapshot(Snapshot)
  - CreateCacheCluster(CacheCluster)
  - CreateCacheParameterGroup(CacheParameterGroup)
  - CreateCacheSecurityGroup(CacheSecurityGroup)
  - CreateCacheSubnetGroup(CacheSubnetGroup)
  - CreateReplicationGroup(ReplicationGroup)
  - CreateSnapshot(Snapshot)
  - DeleteCacheCluster(CacheCluster)
  - DeleteReplicationGroup(ReplicationGroup)
  - DeleteSnapshot(Snapshot)
  - DescribeEngineDefaultParameters(EngineDefaults)
  - ModifyCacheCluster(CacheCluster)
  - ModifyCacheSubnetGroup(CacheSubnetGroup)
  - ModifyReplicationGroup(ReplicationGroup)
  - PurchaseReservedCacheNodesOffering(ReservedCacheNode)

- RebootCacheCluster(CacheCluster)
- RevokeCacheSecurityGroupIngress(CacheSecurityGroup)

- Amazon RDS
  - AddSourceIdentifierToSubscription(EventSubscription)
  - AuthorizeDBSecurityGroupIngress(DBSecurityGroup)
  - CopyDBParameterGroup(DBParameterGroup)
  - CopyDBSnapshot(DBSnapshot)
  - CopyOptionGroup(OptionGroup)
  - CreateDBInstance(DBInstance)
  - CreateDBInstanceReadReplica(DBInstance)
  - CreateDBParameterGroup(DBParameterGroup)
  - CreateDBSecurityGroup(DBSecurityGroup)
  - CreateDBSnapshot(DBSnapshot)
  - CreateDBSubnetGroup(DBSubnetGroup)
  - CreateEventSubscription(EventSubscription)
  - CreateOptionGroup(OptionGroup)
  - DeleteDBInstance(DBInstance)
  - DeleteDBSnapshot(DBSnapshot)
  - DeleteEventSubscription(EventSubscription)
  - DescribeEngineDefaultParameters(EngineDefaults)
  - ModifyDBInstance(DBInstance)
  - ModifyDBSubnetGroup(DBSubnetGroup)
  - ModifyEventSubscription(EventSubscription)
  - ModifyOptionGroup(OptionGroup)
  - PromoteReadReplica(DBInstance)
  - PurchaseReservedDBInstancesOffering(ReservedDBInstance)
  - RebootDBInstance(DBInstance)
  - RemoveSourceIdentifierFromSubscription(EventSubscription)
  - RestoreDBInstanceFromDBSnapshot(DBInstance)
  - RestoreDBInstanceToPointInTime(DBInstance)

    - RevokeDBSecurityGroupIngress(DBSecurityGroup)
- Amazon Redshift
    - AuthorizeClusterSecurityGroupIngress(ClusterSecurityGroup)
    - AuthorizeSnapshotAccess(Snapshot)
    - CopyClusterSnapshot(Snapshot)
    - CreateCluster(Cluster)
    - CreateClusterParameterGroup(ClusterParameterGroup)
    - CreateClusterSecurityGroup(ClusterSecurityGroup)
    - CreateClusterSnapshot(Snapshot)
    - CreateClusterSubnetGroup(ClusterSubnetGroup)
    - CreateEventSubscription(EventSubscription)
    - CreateHsmClientCertificate(HsmClientCertificate)
    - CreateHsmConfiguration(HsmConfiguration)
    - DeleteCluster(Cluster)
    - DeleteClusterSnapshot(Snapshot)
    - DescribeDefaultClusterParameters(DefaultClusterParameters)
    - DisableSnapshotCopy(Cluster)
    - EnableSnapshotCopy(Cluster)
    - ModifyCluster(Cluster)
    - ModifyClusterSubnetGroup(ClusterSubnetGroup)
    - ModifyEventSubscription(EventSubscription)
    - ModifySnapshotCopyRetentionPeriod(Cluster)
    - PurchaseReservedNodeOffering(ReservedNode)
    - RebootCluster(Cluster)
    - RestoreFromClusterSnapshot(Cluster)
    - RevokeClusterSecurityGroupIngress(ClusterSecurityGroup)
    - RevokeSnapshotAccess(Snapshot)
    - RotateEncryptionKey(Cluster)

Enum 클래스가 제거됨

Enum 버전 2에 있던 Aws\S3\Enum\CannedAcl 클래스(예: AWS SDK for PHP)를 제거했습니다. 열거형은 유효한 파라미터 값의 그룹을 나타내는 상수를 포함하는 SDK의 퍼블릭 API 내의 구체적 클래스입니다. 이러한 열거형은 API 버전에 특정하여 시간에 따라 변경되고, PHP 예약 단어와 충돌하여 결국 쓸모없게 되므로 버전 3에서는 제거했습니다. 이는 데이터 기반의 API 버전과 무관한 버전 3의 특성을 지원합니다.

Enum 객체의 값을 사용하는 대신 리터럴 값을 직접 사용합니다(예: CannedAcl::PUBLIC_READ → 'public-read').

세부 예외 클래스가 제거됨

열거형을 제거한 것과 흡사한 이유로 각 서비스의 네임스페이스에 존재했던 세부 예외 클래스(예: Aws\Rds\Exception\{SpecificError}Exception)를 제거했습니다. 서비스 또는 작업에서 발생되는 예외는 사용되는 API 버전에 따라 다릅니다. 즉, 버전에 따라 변경될 수 있습니다. 또한 버전 2의 세부 예외 클래스가 완성되지 않았으므로 지정된 작업에서 발생될 수 있는 전체 예외 목록이 제공되지 않습니다.

따라서 각 서비스에 대한 근본적인 예외 클래스(예: Aws\Rds\Exception\RdsException)를 파악하여 오류를 처리합니다. 예외의 getAwsErrorCode() 메서드를 사용하여 특정 오류 코드를 확인할 수 있습니다. 이는 다른 예외 클래스를 캐치하는 것과 기능적으로 동일하지만, SDK에 부풀림을 추가하지 않고 함수를 제공합니다.

정적 Facade 클래스가 제거됨

AWS SDK for PHP 버전 2에는 enableFacades() 클래스에서 Aws를 호출하여 다양한 서비스 클라이언트에 정적으로 액세스할 수 있도록 하는 기능이 있었습니다. 이 기능은 Laravel에서 영감을 받은 것으로 잘 알려져 있지 않았습니다. 이 기능은 PHP 모범 사례에 위반되므로 1년 전부터 문서화를 중단했습니다. 버전 3에서는 이 기능이 완전히 제거되었습니다. Aws\Sdk 객체에서 클라이언트 객체를 가져온 다음 해당 객체를 정적 클래스가 아닌 객체 인스턴스로 사용합니다.

반복기가 페이지네이터로 대체됨

AWS SDK for PHP 버전 2에는 *반복기*라는 기능이 있습니다. 이 기능은 페이지 지정된 결과를 반복하는 데 사용되는 객체입니다. 이와 관련하여 반복기는 각 결과에서 특정 값만 내보내므로 유연성이 부족하다는 불만이 있었습니다. 따라서 결과에 필요한 다른 값이 있는 경우 이벤트 리스너를 통해 검색해야 합니다.

버전 3에서는 반복기가 페이지네이터로 대체되었습니다. 목적은 비슷하지만 페이지네이터는 더 유연합니다. 이는 응답에서 값 대신 결과 객체를 출력하기 때문입니다.

다음 예에서는 버전 2와 버전 3에서 S3 `ListObjects` 작업에 대한 페이지 지정된 결과를 검색하는 방법을 설명하여 페이지네이터가 반복기와 어떻게 다른지를 보여줍니다.

```
// 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";
    }
}
```

페이지네이터 객체에는 JMEsPath 표현식을 사용하여 결과 집합에서 데이터를 더 쉽게 추출할 수 있는 `search()` 메서드가 있습니다.

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

> **Note**
>
> 버전 3으로 부드럽게 전환할 수 있도록 `getIterator()` 메서드가 계속 지원되지만 코드를 마이그레이션하여 페이지네이터를 사용하는 것이 좋습니다.

많은 상위 수준 추상화가 변경됨

일반적으로 많은 상위 수준 추상화(서비스별 헬퍼 객체, 클라이언트와 별개로)가 개선되거나 업데이트되었습니다. 또한 일부는 제거되었습니다.

- 업데이트 사항:
  - Amazon S3 멀티파트 업로드를 사용하는 방법이 변경되었습니다. Amazon Glacier 멀티파트 업로드도 비슷한 방법으로 변경되었습니다.
  - Amazon S3에 미리 서명된 URL을 생성하는 방법이 변경되었습니다.
  - `Aws\S3\Sync` 네임스페이스가 `Aws\S3\Transfer` 클래스로 대체되었습니다. `S3Client::uploadDirectory()` 및 `S3Client::downloadBucket()` 메서드를 계속 사용할 수 있지만 다른 옵션이 있습니다. PHP용 AWS SDK for PHP 버전 3에서 Amazon S3 Transfer Manager 사용 관련 설명서를 참조하세요.
  - `Aws\S3\Model\ClearBucket` 및 `Aws\S3\Model\DeleteObjectsBatch`가 `Aws\S3\BatchDelete` 및 `S3Client::deleteMatchingObjects()`로 대체되었습니다.
  - AWS SDK for PHP 버전 3에서 DynamoDB 세션 핸들러 사용 옵션과 동작이 약간 변경되었습니다.
  - `Aws\DynamoDb\Model\BatchRequest` 네임스페이스가 `Aws\DynamoDb\WriteRequestBatch`로 대체되었습니다. DynamoDB WriteRequestBatch에 대한 설명서를 참조하세요.
  - 이제 `Aws\Ses\SesClient` 작업을 사용하면 `RawMessage`가 `SendRawEmail`를 인코딩하는 base64를 처리합니다.
- 제거된 항목:
  - Amazon DynamoDB`Item`, `Attribute`, 및 `ItemIterator` 클래스 - 이미 버전 2.7.0부터 사용되지 않습니다.
  - Amazon SNS 메시지 검사기 - 지금은 SDK를 종속 항목으로 요구하지 않는 별도의 경량 프로젝트입니다. 하지만 이 프로젝트는 SDK의 Phar 및 ZIP 배포에 포함됩니다. AWS PHP 개발 블로그에서 시작 안내서를 확인할 수 있습니다.
  - Amazon S3`AcpBuilder` 및 관련 객체를 제거했습니다.

두 SDK 버전의 코드 샘플 비교

다음 예시는 AWS SDK for PHP 버전 3의 사용 방식이 버전 2와 어떻게 다른지 보여줍니다.

예: Amazon S3 **`ListObjects`** 작업

SDK 버전 2

```
<?php
```

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

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

$s3 = S3Client::factory([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

SDK 버전 3

주요 차이점:

- new 대신 `factory()`를 사용하여 클라이언트를 인스턴스화합니다.
- 인스턴스화 중에 `'version'` 및 `'region'` 옵션이 필요합니다.

```
<?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";
}
```

예: 글로벌 구성으로 클라이언트 인스턴스화

SDK 버전 2

```
<?php return array(
    'includes' => array('_aws'),
    'services' => array(
        'default_settings' => array(
            'params' => array(
                'profile' => 'my_profile',
                'region'  => 'us-east-1'
            )
        ),
        'dynamodb' => array(
            'extends' => 'dynamodb',
            'params' => array(
                'region'  => 'us-west-2'
            )
        ),
    )
);
```

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\Common\Aws;
```

```
$aws = Aws::factory('path/to/my/config.php');

$sqs = $aws->get('sqs');
// Note: SQS client will be configured for us-east-1.

$dynamodb = $aws->get('dynamodb');
// Note: DynamoDB client will be configured for us-west-2.
```

SDK 버전 3

주요 차이점:

- `Aws\Sdk` 대신 `Aws\Common\Aws` 클래스를 사용합니다.
- 구성 파일이 없습니다. 구성에 대한 배열을 대신 사용합니다.
- 인스턴스화 중에 `'version'` 옵션이 필요합니다.
- `create<Service>()` 대신 `get('<service>')` 메서드를 사용합니다.

```
<?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.
```

에 대한 보안 AWS SDK for PHP

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다. 보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 AWS - 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 AWS 제공할 책임이 있습니다. 당사의 보안 책임은에서 최우선 순위이며 AWS, 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안의 효과를 정기적으로 테스트하고 검증합니다.

클라우드의 보안 - 사용자의 책임은 사용 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항 및 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

주제

- AWS SDK for PHP 버전 3의 데이터 보호
- 자격 증명 및 액세스 관리
- 이 AWS 제품 또는 서비스에 대한 규정 준수 검증
- 이 AWS 제품 또는 서비스에 대한 복원력
- 이 AWS 제품 또는 서비스에 대한 인프라 보안
- AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)
- AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로)

AWS SDK for PHP 버전 3의 데이터 보호

AWS 공동 책임 모델의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사

용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 AWS SDK for PHP 버전 3 또는 기타 AWS 서비스 로 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- IAM AWS 서비스 작업 방법

- AWS 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은에서 수행하는 작업에 따라 다릅니다 AWS.

서비스 사용자 - AWS 서비스 를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. 에서 기능에 액세스할 수 없는 경우 사용 중인의 AWS 자격 증명 및 액세스 문제 해결 또는 사용 설명서를 AWS참조 AWS 서비스 하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 경우에 대한 전체 액세스 권한을 가지고 있을 것입니다 AWS. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 IAM을 사용하는 방법에 대해 자세히 알아보려면 사용 중인의 AWS 서비스 사용 설명서를 AWS참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 자격 AWS 증명 기반 정책 예제를 보려면 사용 중인의 사용 설명서를 참조 AWS 서비스 하세요.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용자 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하

지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용자 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 IAM 사용 설명서의 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용자 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용자 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용자 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용자 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

자격 증명 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용자 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

IAM AWS 서비스 작업 방법

가 대부분의 IAM 기능을 AWS 서비스 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

IAM AWS 서비스 에서 특정를 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 섹션을 참조하세요.

AWS 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단 AWS 하고 수정할 수 있습니다.

주제

- 에서 작업을 수행할 권한이 없음 AWS
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없음 AWS

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 awes:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

이 경우, awes:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 에서 이러한 기능을 AWS 지원하는지 여부를 알아보려면 섹션을 참조하세요IAM AWS 서비스 작업 방법.

- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용자 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용자 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 규정 준수 검증

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 작업 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.

AWS 리전 는 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다.

가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 작업 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 인프라 보안

이 AWS 제품 또는 서비스는 관리형 서비스를 사용하므로 글로벌 네트워크 보안으로 AWS 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해이 AWS 제품 또는 서비스에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)

> **Note**
>
> Amazon S3 암호화 클라이언트 버전 2(V2)를 사용하고 버전 3(V3)으로 마이그레이션하려는 경우 섹션을 참조하세요AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로).

이 주제에서는 Amazon Simple Storage Service(S3) 암호화 클라이언트 버전 1(V1) 에서 버전 2(V2)로 애플리케이션을 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 보여줍니다.

마이그레이션 개요

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트하세요. 먼저, AWS SDK for PHP 의 업데이트된 버전을 애플리케이션에 배포합니다. 이렇게 하면 기존 V1 암호화 클라이언트가 새 V2 클라이언트가 작성한 객체를 해독할 수 있습니다. 애플리케이션에서 다중 AWS SDKs 사용하는 경우 각 SDK를 별도로 업그레이드해야 합니다.

2. 암호화 및 복호화 클라이언트를 V2로 마이그레이션합니다. 모든 V1 암호화 클라이언트가 새 형식을 읽을 수 있게 되면 기존 암호화 및 복호화 클라이언트를 각각의 V2 버전으로 마이그레이션할 수 있습니다.

새 형식을 읽도록 기존 클라이언트를 업데이트하세요

V2 암호화 클라이언트는 이전 버전의 클라이언트에서 지원하지 않는 암호화 알고리즘을 사용합니다. 마이그레이션의 첫 번째 단계는 V1 복호화 클라이언트를 최신 SDK 릴리스로 업데이트하는 것입니다. 이 단계를 완료하면 애플리케이션의 V1 클라이언트가 V2 암호화 클라이언트로 암호화된 객체를 해독할 수 있습니다. AWS SDK for PHP의 각 주요 버전에 대한 세부 정보는 아래를 참조하세요.

AWS SDK for PHP 버전 3 업그레이드

버전 3은 AWS SDK for PHP의 최신 버전이 아닙니다. 이 마이그레이션을 완료하려면 패키지 버전 3.148.0 이상을 사용해야 합니다. `aws/aws-sdk-php`

명령줄에서 설치

Composer를 사용하여 설치한 프로젝트의 경우 Composer 파일에서 SDK 패키지를 SDK 버전 3.148.0으로 업데이트한 후 다음 명령을 실행합니다.

```
composer update aws/aws-sdk-php
```

Phar 또는 ZIP 파일을 사용하여 설치

다음 방법 중 하나를 사용합니다. 업데이트된 SDK 파일은 require 문에 의해 결정되는 코드에서 요구하는 위치에 배치해야 합니다.

Phar 파일을 사용하여 설치한 프로젝트의 경우 업데이트된 파일을 다운로드하세요. aws.phar

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

Zip 파일을 사용하여 설치한 프로젝트의 경우 업데이트된 파일을 다운로드하세요.

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

암호화 및 암호 해독 클라이언트를 V2로 마이그레이션

새 암호화 형식을 읽도록 클라이언트를 업데이트한 후 애플리케이션을 V2 암호화 및 복호화 클라이언트로 업데이트할 수 있습니다. 다음 단계는 V1에서 V2로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다.

V2 클라이언트로 업데이트하기 위한 요구 사항

1. AWS KMS 암호화 컨텍스트는 `S3EncryptionClientV2::putObject` 및 `S3EncryptionClientV2::putObjectAsync` 메서드에 전달되어야 합니다. AWS KMS 암호화 컨텍스트는 키 암호화를 위해 암호화 컨텍스트에 추가해야 하는 AWS KMS 키-값 페어의 결합 배열입니다. 추가 컨텍스트가 필요하지 않은 경우 빈 배열을 전달할 수 있습니다.

2. `@SecurityProfile`은 `S3EncryptionClientV2`의 `getObject` 및 `getObjectAsync` 메서드에 전달되어야 합니다. `@SecurityProfile`은 `getObject...` 메서드의 새로운 필수 파라미터입니다. `'V2'`로 설정하면 V2 호환 형식으로 암호화된 객체만 해독할 수 있습니다. 이 파라미터를 `'V2_AND_LEGACY'`로 설정하면 V1 호환 형식으로 암호화된 객체도 해독할 수 있습니다. 마이그레이션을 지원하려면 `@SecurityProfile`을 `'V2_AND_LEGACY'`로 설정합니다. 새 애플리케이션 개발에만 `'V2'`를 사용하세요.

3. (선택 사항) `S3EncryptionClientV2::getObject`에 `@KmsAllowDecryptWithAnyCmk` 파라미터를 포함시키고 `@KmsAllowDecryptWithAnyCmk` 호출에 `S3EncryptionClientV2::getObjectAsync* methods.` 새 파라미터가 추가되었습니다. KMS 키를 제공하지 않고도 암호 해독이 가능하도록 이 파라미터를 `true`에 설정합니다. 기본값은 `false`입니다.

4. V2 클라이언트를 사용한 암호 해독의 경우 `@KmsAllowDecryptWithAnyCmk` 파라미터가 `"getObject..."` 메서드 호출에 대해 `true`에 설정되지 않은 경우 `KmsMaterialsProviderV2` 생성자에 `kms-key-id`를 제공해야 합니다.

마이그레이션 예제

예 1: V2 클라이언트로 마이그레이션

사전 마이그레이션

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

마이그레이션 후

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

예제 2: kms-key-id와 AWS KMS 함께 사용

> **Note**
>
> 이 예제에서는 예제 1에 정의된 가져오기와 변수를 사용합니다. 예를 들어 `$encryptionClient`입니다.

사전 마이그레이션

```
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,
]);
```

마이그레이션 후

```
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
```

```
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로)

> **Note**
>
> Amazon S3 암호화 클라이언트 버전 1(V1)을 사용하는 경우 버전 3(V3)으로 마이그레이션하기 전에 먼저 버전 2(V2)로 마이그레이션해야 합니다. AWS SDK for PHP 버전 3의 Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)을(를) 참조하세요.

이 주제에서는 Amazon Simple Storage Service(Amazon S3) 암호화 클라이언트 버전 2(V2)에서 버전 3(V3)으로 애플리케이션을 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 보여줍니다. 버전 3에는 보안을 강화하고 데이터 키 변조로부터 보호하기 위해 키 커밋 및 커밋 정책이 포함된 AES GCM이 도입되었습니다.

마이그레이션 개요

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트하세요. 먼저, AWS SDK for PHP 의 업데이트된 버전을 애플리케이션에 배포합니다. 이렇게 하면 기존 V2 암호화 클라이언트가 새 V3 클라이언트가 작성한 객체를 해독할 수 있습니다. 애플리케이션에서 다중 AWS SDKs 사용하는 경우 각 SDK를 별도로 업그레이드해야 합니다.

2. 암호화 및 복호화 클라이언트를 V3로 마이그레이션합니다. 모든 V2 암호화 클라이언트가 새 형식을 읽을 수 있게 되면 기존 암호화 및 복호화 클라이언트를 해당 V3 버전으로 마이그레이션할 수 있습니다.

V3 개념 이해

Amazon S3 암호화 클라이언트 버전 3에는 커밋 정책과 키 커밋 알고리즘이 포함된 AES GCM이라는 두 가지 주요 보안 개선 사항이 도입되었습니다. 성공적인 마이그레이션을 위해서는 이러한 개념을 이해하는 것이 필수적입니다.

약정 정책

커밋 정책은 암호화 및 복호화 작업 중에 암호화 클라이언트가 키 커밋을 처리하는 방법을 제어합니다. 버전 3은 세 가지 정책 옵션을 제공합니다.

`FORBID_ENCRYPT_ALLOW_DECRYPT`

암호화 동작: 키 커밋 없이 객체를 암호화합니다.

복호화 동작: 키 커밋을 사용하거나 사용하지 않고 암호화된 객체의 복호화를 허용합니다.

보안 영향:이 정책은 새로 암호화된 객체에 키 커밋을 적용하지 않으므로 데이터 키 변조가 허용될 수 있습니다. V2 클라이언트와의 호환성을 유지해야 하는 초기 마이그레이션 단계에서만이 정책을 사용합니다.

버전 호환성:이 정책으로 암호화된 객체는 모든 V2 및 V3 구현에서 읽을 수 있습니다.

`REQUIRE_ENCRYPT_ALLOW_DECRYPT`

암호화 동작: `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 알고리즘을 사용하여 키 커밋으로 객체를 암호화합니다.

복호화 동작: 키 커밋을 사용하거나 사용하지 않고 암호화된 객체의 복호화를 허용합니다.

보안 영향:이 정책은 기존 객체를 읽을 수 있는 기능을 유지하면서 새로 암호화된 객체에 대한 보안을 강화합니다. 이는 대부분의 마이그레이션 시나리오에 권장되는 정책입니다.

버전 호환성:이 정책으로 암호화된 객체는 V3 및 최신 V2 구현에서만 읽을 수 있습니다.

마이그레이션 고려 사항:이 정책을 사용하기 전에 암호화된 객체를 읽어야 하는 모든 클라이언트가 V3 또는 최신 V2로 업그레이드되었는지 확인합니다.

`REQUIRE_ENCRYPT_REQUIRE_DECRYPT`

암호화 동작: `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 알고리즘을 사용하여 키 커밋으로 객체를 암호화합니다.

복호화 동작: 키 커밋으로 암호화된 객체의 복호화만 허용합니다. 키 커밋 없이 암호화된 객체는 복호화에 실패합니다.

보안 영향:이 정책은 암호화 및 복호화 모두에 대한 키 커밋을 적용하여 최고 수준의 보안을 제공합니다. 키 커밋을 사용하도록 모든 객체를 마이그레이션한 후에만이 정책을 사용합니다.

버전 호환성: V3 구현만이 정책을 사용할 수 있습니다. 이 정책을 사용하여 V1 또는 V2 암호화된 객체를 복호화하려고 하면 실패합니다.

마이그레이션 고려 사항:이 정책은 전체 마이그레이션을 완료하고 키 커밋으로 기존 객체를 모두 다시 암호화한 후에만 사용해야 합니다.

키 커밋이 있는 AES GCM

키 커밋이 포함된 AES GCM(ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY) 알고리즘은 V3에 도입된 새로운 암호화 알고리즘으로, 데이터 키 변조 공격으로부터 보호합니다.

보안 강화: 데이터 키를 암호화된 콘텐츠에 암호화 방식으로 바인딩하여 데이터 키 변조를 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY 방지합니다. 이렇게 하면 공격자가 복호화 중에 다른 데이터 키를 대체하지 못하여 의도하지 않은 데이터가 복호화될 수 있습니다.

버전 호환성: 로 암호화된 객체는 V3 및 Amazon S3 암호화 클라이언트의 최신 V2 구현에서만 해독할 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY 수 있습니다. V1 클라이언트는이 알고리즘으로 암호화된 객체를 해독할 수 없습니다.

> **Important**
>
> 업그레이드 요구 사항: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY (REQUIRE_ENCRYPT_ALLOW_DECRYPT 또는 REQUIRE_ENCRYPT_REQUIRE_DECRYPT 정책을 사용하여)를 사용하여 암호화를 활성화하기 전에 암호화된 객체를 읽어야 하는 모든 클라이언트가 V3로 업그레이드되었는지 확인해야 합니다. 모든 리더를 업그레이드하지 않으면 키 커밋으로 암호화된 객체에 대한 복호화 실패가 발생합니다.

새 형식을 읽도록 기존 클라이언트를 업데이트하세요

V3 암호화 클라이언트는 이전 버전의 클라이언트가 지원하지 않는 암호화 알고리즘과 키 커밋 기능을 사용합니다. 마이그레이션의 첫 번째 단계는 V2 복호화 클라이언트를 최신 SDK 릴리스로 업데이트하는 것입니다. 이 단계를 완료하면 애플리케이션의 V2 클라이언트가 V3 암호화 클라이언트로 암호화된 객체를 해독할 수 있습니다. 의 각 설치 방법에 대한 세부 정보는 아래를 참조하세요 AWS SDK for PHP.

최신 SDK 버전 빌드 및 설치

이 마이그레이션을 완료하려면 V3 암호화 클라이언트 지원이 포함된 aws/aws-sdk-php 패키지의 최신 버전을 사용해야 합니다.

Composer에서 설치

Composer를 사용하여 설치된 프로젝트의 경우 Composer 파일에서 SDK 패키지를 최신 버전의 SDK로 업데이트한 다음 다음 명령을 실행합니다.

```
composer update aws/aws-sdk-php
```

Phar 또는 ZIP 파일을 사용하여 설치

다음 방법 중 하나를 사용합니다. 업데이트된 SDK 파일은 require 문에 의해 결정되는 코드에서 요구하는 위치에 배치해야 합니다.

Phar 파일을 사용하여 설치한 프로젝트의 경우 업데이트된 파일을 다운로드하세요. aws.phar

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

Zip 파일을 사용하여 설치한 프로젝트의 경우 업데이트된 파일을 다운로드하세요.

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

애플리케이션 구축, 설치 및 배포

SDK를 업데이트한 후 애플리케이션을 다시 빌드하고 배포하여 모든 구성 요소가 업데이트된 버전을 사용하고 있는지 확인합니다. 이 단계는 V2 클라이언트가 V3 클라이언트로 암호화된 객체를 읽을 수 있도록 하는 데 매우 중요합니다.

조직의 표준 배포 절차에 따라 업데이트된 애플리케이션을 롤아웃합니다. 암호화 및 복호화 클라이언트를 V3로 마이그레이션하기 전에 애플리케이션의 모든 인스턴스가 업데이트되었는지 확인합니다.

배포 후 애플리케이션이 여전히 기존 객체를 해독할 수 있고 정상 작업 중에 오류가 발생하지 않는지 확인합니다. 이렇게 하면 SDK 업데이트가 성공했으며 애플리케이션이 다음 마이그레이션 단계를 수행할 준비가 되었음을 확인할 수 있습니다.

암호화 및 복호화 클라이언트를 V3로 마이그레이션

새 암호화 형식을 읽도록 클라이언트를 업데이트한 후 애플리케이션을 V3 암호화 및 복호화 클라이언트로 업데이트할 수 있습니다. 다음 예제에서는 V2에서 V3로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다.

V3 암호화 클라이언트 사용

V3는 `S3EncryptionClientV3` 클래스 및를 도입`KmsMaterialsProviderV3`하여 V2에 상응하는 클래스를 대체합니다. V3의 주요 차이점은 다음과 같습니다.

- V3는 `KmsMaterialsProviderV3` (V2와 동일)를 사용하지만 `GetObject` 호출에서 객체를 해독할 때 암호화 컨텍스트를 확인합니다.
- V3는 암호화 및 복호화 동작을 제어하는 커밋 정책을 도입합니다.

예: KMS 암호화를 사용하여 V2에서 V3로 마이그레이션

마이그레이션 전(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,
]);
```

마이그레이션 중(이전 버전과의 호환성을 갖춘 V3)

```
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,
]);
```

마이그레이션 후(키 커밋이 포함된 V3)

```
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,
]);
```

V3의 주요 차이점:

- KmsMaterialsProviderV2 대신 KmsMaterialsProviderV3 사용
- @KmsEncryptionContext 파라미터는 putObject 여전히 작업에 필요합니다.
- @KmsEncryptionContext 파라미터는 getObject 작업에 대해 선택 사항이며 제공된 암호화 컨텍스트가 객체의 컨텍스트와 일치하는지 확인합니다.
- @SecurityProfile 파라미터는 해독할 수 있는 암호화 버전을 제어합니다. 마이그레이션 중에 V1 및 V2로 암호화된 객체 읽기'V3_AND_LEGACY'를 지원하려면 로 설정
- @CommitmentPolicy 파라미터는이 작업에 대한 커밋 정책을 제어합니다. 마이그레이션 중에 커밋되지 않은 암호화된 객체를 읽을 'FORBID_ENCRYPT_ALLOW_DECRYPT' 수 있도록 로 설정

추가 예제

다음 예제에서는 마이그레이션 프로세스를 관리하고 암호화 동작을 제어하는 데 도움이 되는 V3에서 사용할 수 있는 추가 구성 옵션을 보여줍니다.

레거시 지원 사용

마이그레이션 중에 Amazon S3 암호화 클라이언트의 V1 또는 V2로 암호화된 객체를 복호화해야 할 수 있습니다. @SecurityProfile 파라미터는 V3 클라이언트가 해독할 수 있는 암호화 버전을 제어합니다.

이 구성을 사용해야 하는 경우: 애플리케이션이 V1 또는 V2 클라이언트로 암호화된 객체를 읽어야 하는 경우 'V3_AND_LEGACY' 보안 프로필을 사용합니다. 이는 버킷에 이전 암호화된 객체와 새 암호화된 객체가 혼합되어 있는 마이그레이션 기간 동안 흔히 발생합니다.

```
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,
]);
```

`@SecurityProfile` 파라미터는 다음 값 중 하나를 받습니다.

- `'V3'` (기본값): 키 커밋을 사용하여 V3로 암호화된 객체만 복호화
- `'V3_AND_LEGACY'`: V1, V2 또는 V3로 암호화된 객체 복호화

> **Important**
>
> 마이그레이션을 완료하고 V3를 사용하여 모든 객체를 다시 암호화한 후에는 `@SecurityProfile` 파라미터를 제거하거나 로 설정`'V3'`하여 보안을 극대화해야 합니다.

스토리지 방법 구성

Amazon S3 암호화 클라이언트는 객체의 메타데이터 헤더 또는 별도의 명령 파일이라는 두 가지 방법으로 암호화 메타데이터를 저장할 수 있습니다. @MetadataStrategy 파라미터는 사용되는 스토리지 방법을 제어합니다.

이 구성을 사용하는 경우: 원본 객체 메타데이터를 보존'INSTRUCTION_FILE'해야 하거나 메타데이터 크기 제약이 있는 객체로 작업할 때 사용합니다. 객체와 함께 암호화 메타데이터를 저장할 수 있는 더 간단한 배포에는 'METADATA' (기본값)를 사용합니다.

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

`@MetadataStrategy` 파라미터는 다음 값 중 하나를 받습니다.

- `'METADATA'` (기본값): 객체의 메타데이터 헤더에 암호화 메타데이터 저장
- `'INSTRUCTION_FILE'`: 암호화 메타데이터를 접미사가 있는 별도의 명령 파일에 저장 `.instruction`

> **Note**
>
> `'INSTRUCTION_FILE'`를 사용할 때 키 커밋 알고리즘이 포함된 AES GCM은 데이터 키 변조에 대한 추가 보호를 제공합니다. `'METADATA'` 스토리지를 사용하는 객체는이 추가 보호의 이점을 얻지 못합니다.

AWS SDK for PHP 버전 3에 대한 FAQ

클라이언트에서 사용 가능한 메서드는 무엇입니까?

AWS SDK for PHP는 서비스 설명 및 동적 magic __call() methods를 사용하여 API 작업을 실행합니다. 웹 서비스 클라이언트에 사용 가능한 전체 메서드 목록은 클라이언트의 API 설명서에서 확인할 수 있습니다.

cURL SSL 인증서 오류가 발생한 경우 어떻게 해야 하나요?

이 문제는 cURL 및 SSL을 포함하는 최신이 아닌 CA 번들을 사용할 때 발생할 수 있습니다. 서버에서 CA 번들을 업데이트하거나 cURL 웹 사이트에서 직접 최신 CA 번들을 다운로드하여 이 문제를 해결할 수 있습니다.

기본적으로 AWS SDK for PHP는 PHP를 컴파일할 때 구성되는 CA 번들을 사용합니다. `openssl.cafile` PHP .ini 구성 설정을 디스크에 있는 CA 파일의 경로로 설정하도록 수정하여 PHP에서 사용되는 기본 CA 번들을 변경할 수 있습니다.

클라이언트에서 사용 가능한 API 버전은 무엇입니까?

`version` 옵션은 클라이언트를 생성할 때 필요합니다. 사용 가능한 API 버전 목록은 각 클라이언트의 API 설명서 페이지(::aws-php-class:<index.html>)에서 확인할 수 있습니다. 특정 API 버전을 로드할 수 없는 경우 AWS SDK for PHP의 사본을 업데이트해야 할 수도 있습니다.

클라이언트의 API 공급자가 찾을 수 있는 최신 API 버전을 사용하려면 `latest` 문자열을 "버전" 구성 값에 제공합니다(기본 api_provider는 API용 SDK 모델의 `src/data` 디렉터리를 스캔함).

> **Warning**
>
> API 업데이트를 포함하는 SDK의 새 마이너 버전을 끌어오면 프로덕션 애플리케이션이 중단될 수 있으므로 프로덕션 애플리케이션에서 `latest`를 사용하지 않는 것이 좋습니다.

클라이언트에서 사용 가능한 리전 버전은 무엇입니까?

`region` 옵션은 클라이언트를 생성할 때 필요하며, 문자열 값을 사용하여 지정합니다. 사용할 수 있는 AWS 리전 목록은 AWS 일반 참조의 AWS 리전 및 엔드포인트를 참조하세요.

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

2GB보다 큰 파일은 왜 업로드하거나 다운로드할 수 없나요?

PHP는 부호가 있는 정수 유형을 사용하는데 대부분의 플랫폼에서는 32비트 정수를 사용하므로, AWS SDK for PHP는 32비트 스택에서 2GB보다 큰 파일을 올바르게 처리하지 못합니다. 여기서 "스택"은 CPU, OS, 웹 서버, PHP 바이너리를 포함합니다. 이것은 잘 알려진 PHP 문제입니다. Microsoft Windows의 경우 PHP 7 빌드에서만 64비트 정수를 지원합니다.

권장 솔루션은 최신 버전의 PHP가 설치된 64비트 Linux 스택(예: 64비트 Amazon Linux AMI)을 사용하는 것입니다.

자세한 내용은 PHP filesize: Return values 관련 문서를 참조하세요.

네트워크를 통해 전송되는 데이터를 확인하려면 어떻게 해야 하나요?

클라이언트 생성자에서 debug 옵션을 사용하여 네트워크를 통해 전송되는 데이터를 비롯한 디버그 정보를 가져올 수 있습니다. 이 옵션을 true로 설정하면 실행 중인 명령의 모든 변형, 전송 중인 요청, 수신 중인 응답, 처리 중인 결과가 STDOUT로 방출됩니다. 여기에는 네트워크를 통해 주고 받는 데이터가 포함됩니다.

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

요청에 대한 임의 헤더를 설정하려면 어떻게 해야 하나요?

Aws\HandlerList 또는 Aws\CommandInterface의 Aws\ClientInterface에 사용자 지정 미들웨어를 추가하여 서비스 작업에 임의 헤더를 추가할 수 있습니다. 다음 예에서는 Aws

`\Middleware::mapRequest` 헬퍼 메서드를 사용하여 특정 Amazon S3`PutObject` 작업에 `X-Foo-Baz` 헤더를 추가하는 방법을 보여줍니다.

자세한 내용은 mapRequest를 참조하세요.

임의 요청에 서명하려면 어떻게 해야 하나요?

SDK:AWS-PHP-Class: SignatureV4 클래스 <class-Aws.Signature.SignatureV4.html>를 사용하여 임의::aws-php-class: PSR-7 요청 <class-Psr.Http.Message.RequestInterface.html>에 서명할 수 있습니다.

이렇게 수행하는 방법에 대한 전체 예제는 AWS SDK for PHP 버전 3으로 사용자 지정 Amazon CloudSearch 도메인 요청에 서명을 참조하세요.

명령을 전송하기 전에 수정하려면 어떻게 해야 하나요?

`Aws\HandlerList` 또는 `Aws\CommandInterface`의 `Aws\ClientInterface`에 사용자 지정 미들웨어를 추가하여 명령을 전송하기 전에 수정할 수 있습니다. 다음 예에서는 명령을 전송하기 전에 명령에 사용자 지정 명령 파라미터를 추가하는 방법을 보여줍니다. 이때 기본 옵션을 추가해야 합니다. 이 예에서는 `Aws\Middleware::mapCommand` 헬퍼 메서드를 사용합니다.

자세한 내용은 mapCommand를 참조하세요.

CredentialsException이란 무엇입니까?

`Aws\Exception\CredentialsException`를 사용하는 동안 AWS SDK for PHP이 표시되는 경우, SDK에 자격 증명이 제공되지 않아 사용 중인 환경에서 자격 증명을 찾을 수 없다는 것을 의미합니다.

자격 증명 없이 클라이언트를 인스턴스화한 경우 서비스 작업을 처음으로 수행할 때 SDK에서 자격 증명을 찾으려고 시도합니다. 먼저 일부 특정 환경 변수를 확인한 다음 구성된 Amazon EC2 인스턴스에서만 사용 가능한 인스턴스 프로파일 자격 증명을 찾습니다. 자격 증명이 제공되지 않았거나 없는 경우 `Aws\Exception\CredentialsException`이 발생합니다.

이 오류가 표시되는 경우 인스턴스 프로파일 자격 증명을 사용하려면 SDK가 실행 중인 Amazon EC2 인스턴스가 적절한 IAM 역할로 구성되어 있는지 확인해야 합니다.

이 오류가 표시되는 경우 인스턴스 프로파일 자격 증명을 사용하지 않으려면 SDK에 자격 증명을 올바르게 제공했는지 확인해야 합니다.

자세한 내용은 AWS SDK for PHP 버전 3의 자격 증명을 참조하세요.

AWS SDK for PHP는 HHVM에서 작동합니까?

AWS SDK for PHP는 현재 HHVM에서 실행되지 않으며, HHVM의 배포량 시맨틱 문제가 해결될 때까지 사용할 수 없습니다.

SSL을 비활성화하려면 어떻게 해야 하나요?

클라이언트 팩토리 메서드의 `scheme` 파라미터를 'http'로 설정하여 SSL을 비활성화할 수 있습니다. 모든 서비스에서 `http` 액세스를 지원하는 것은 아닙니다. 모든 지원 리전 및 엔드포인트 목록은 AWS 일반 참조에서 AWS 리전 및 엔드포인트를 참조하세요.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

> **Warning**
>
> SSL에서 모든 데이터를 암호화해야 하고, 연결 핸드셰이크를 완료하는 데 TCP보다 더 많은 TCP 패킷이 필요하므로 SSL을 비활성화하면 성능이 약간 향상될 수 있습니다. 하지만 SSL을 비활성화하면 모든 데이터가 암호화되지 않은 네트워크를 통해 전송됩니다. 따라서 SSL을 비활성화하기 전에 보안 영향과 네트워크를 통한 염탐 가능성을 신중하게 고려해야 합니다.

"구문 분석 오류"가 발생한 경우 어떻게 해야 하나요?

PHP 엔진은 이해할 수 없는 구문이 발견될 경우 구문 분석 오류를 발생합니다. 이 오류는 다른 버전의 PHP용으로 작성된 코드를 실행하려고 시도할 때 거의 항상 발생합니다.

구문 분석 오류가 발생하는 경우, 시스템이 SDK의 AWS SDK for PHP 버전 3에 대한 요구 사항 및 권장 사항을 충족하는지 확인해야 합니다.

Amazon S3 클라이언트가 gzip으로 압축된 파일을 해제하는 이유는 무엇입니까?

기본 Guzzle 6 HTTP 핸들러를 비롯한 일부 HTTP 핸들러에서는 압축된 응답 본문을 기본적으로 압축 해제합니다. decode_content HTTP 옵션을 `false`로 설정하여 이 동작을 재정의할 수 있습니다. 이전

버전과의 호환성을 위해 이 기본값은 변경할 수 없지만, S3 클라이언트 수준에서 콘텐츠 디코딩을 비활성화하는 것이 좋습니다.

자동 콘텐츠 디코딩을 비활성화하는 방법을 보여주는 예는 decode_content를 참조하세요.

Amazon S3에서 본문 서명을 비활성화하려면 어떻게 해야 하나요?

명령 객체의 `ContentSHA256` 파라미터를 `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`로 설정하여 본문 서명을 비활성화할 수 있습니다. 그러면 AWS SDK for PHP는 이 파라미터를 'x-amz-content-sha-256' 헤더로 사용하고 표준 요청의 본문 체크섬을 사용합니다.

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

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
    'ContentSHA256' => Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD
];

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

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

AWS SDK for PHP는 재시도 스키마를 어떻게 처리하나요?

AWS SDK for PHP에는 재시도 동작을 처리하는 `RetryMiddleware`가 있습니다. 서버 오류에 대한 5xx HTTP 상태 코드 측면에서 SDK는 500, 502, 503 및 504에 대해 재시도됩니다.

재시도를 통해 `RequestLimitExceeded`, `Throttling`, `ProvisionedThroughputExceededException`, `ThrottlingException`, `RequestThrottled`, `BandwidthLimitExceeded`를 비롯한 예외를 처리합니다.

또한 AWS SDK for PHP는 지수 지연을 재시도 스키마의 백오프 및 지터 알고리즘과 통합합니다. 모든 서비스에 대한 기본 재시도 동작은 3으로 구성되며, Amazon DynamoDB만 예외적으로 10으로 구성됩니다.

오류 코드가 있는 예외를 처리하려면 어떻게 해야 하나요?

AWS SDK for PHP 사용자 지정 `Exception` 클래스 외에 각 AWS 서비스 클라이언트에는 AwsExceptionAwsException에서 상속되는 자체 예외 클래스가 있습니다. 각 메서드의 `Errors` 섹션에 나열된 API 관련 오류를 포착하기 위해 오류 유형을 자세히 결정할 수 있습니다.

오류 코드 정보는 `Aws\Exception\AwsException`에서 getAwsErrorCode()로 확인할 수 있습니다.

```
$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;
    }
}
```

용어집

API 버전

서비스에는 하나 이상의 API 버전이 있습니다. 사용 중인 버전에 따라 유효한 작업 및 파라미터가 결정됩니다. API 버전은 날짜와 비슷하게 형식 지정됩니다. 예를 들어, Amazon S3의 최신 API 버전은 `2006-03-01`입니다. 클라이언트 객체를 구성할 때 버전을 지정합니다.

클라이언트

클라이언트 객체는 서비스에 대한 작업을 실행하는 데 사용됩니다. SDK에서 지원되는 각 서비스에는 해당 클라이언트 객체가 있습니다. 클라이언트 객체에는 서비스 작업과 일대일로 대응하는 메서드가 있습니다. 클라이언트 사용을 시작하려면 the section called “기본 서비스 클라이언트 생성” 섹션을 참조하세요.

Command

명령 객체는 작업 실행을 캡슐화합니다. 요청 만들기 섹션에 설명된 대로 서비스 작업을 실행할 때 명령 객체를 직접 처리하지는 않습니다. 동시 요청, 일괄 처리 등과 같은 SDK의 고급 기능을 사용하려면 클라이언트의 `getCommand()` 메서드를 사용하여 명령 객체에 액세스할 수 있습니다. 자세한 내용은 AWS SDK for PHP 버전 3의 명령 객체 가이드를 참조하세요.

핸들러

핸들러는 명령 및 요청을 결과로 실제로 변환하는 함수입니다. 핸들러는 일반적으로 HTTP 요청을 전송합니다. 핸들러를 미들웨어와 함께 구성하여 동작을 강화할 수 있습니다. 핸들러는 `Aws\CommandInterface` 및 `Psr\Http\Message\RequestInterface`를 받아 `Aws\ResultInterface`와 함께 실행되거나 `Aws\Exception\AwsException` 이유와 함께 거부되는 promise를 반환하는 함수입니다.

JMESPath

JMESPath는 JSON과 유사한 데이터에 대한 쿼리 언어입니다. AWS SDK for PHP는 JMESPath 표현식을 사용하여 PHP 데이터 구조를 쿼리합니다. `Aws\Result` 메서드를 통해 `Aws\ResultPaginator` 및 `search($expression)` 객체에 대해 JMESPath 표현식을 직접 사용할 수 있습니다.

미들웨어

미들웨어는 특수 유형의 상위 레벨 함수로서, 명령을 전송하는 동작을 강화하고 “다음” 핸들러에 위임합니다. 미들웨어 함수는 `Aws\CommandInterface` 및 `Psr\Http\Message`

`\RequestInterface`를 받아 `Aws\ResultInterface`와 함께 실행되거나 `Aws\Exception\AwsException` 이유와 함께 거부되는 promise를 반환합니다.

작업

서비스 API의 단일 작업(예: DynamoDB의 경우는 `CreateTable`, Amazon EC2의 경우는 `RunInstances`)을 의미합니다. SDK에서는 해당 서비스의 클라이언트 객체에서 동일한 이름의 메서드를 호출하여 작업을 실행합니다. 작업 실행에는 HTTP 요청을 준비하여 서비스로 보내고 응답을 구문 분석하는 과정이 포함됩니다. 이 작업 실행 프로세스는 SDK에서 명령 객체를 통해 추상화됩니다.

페이지네이터

일부 AWS 서비스 작업은 페이지 지정되며 잘린 결과로 응답합니다. 예를 들어 Amazon S3의 `ListObjects` 작업은 한 번에 최대 1,000개의 객체만 반환합니다. 이러한 작업은 후속 요청에 토큰(마커) 파라미터를 추가해야 전체 결과값 세트를 검색할 수 있습니다. 페이지네이터란 개발자가 페이지 지정 API를 더 쉽게 사용할 수 있도록 이 프로세스의 추상화 역할을 하는 SDK 기능입니다. 이 기능은 클라이언트의 `getPaginator()` 메서드를 통해 액세스합니다. 자세한 내용은 AWS SDK for PHP 버전 3의 페이지네이터 가이드를 참조하세요.

Promise

promise는 비동기 작업의 최종 결과를 나타냅니다. promise와 상호 작용하는 기본 방법은 then 메서드를 통해 수행하는 것입니다. 이 메서드는 promise의 최종 값 또는 promise를 이행할 수 없는 이유를 수신할 콜백을 등록합니다.

리전

서비스는 하나 이상의 지리적 리전에서 지원됩니다. 애플리케이션에서의 데이터 지연 시간을 줄이기 위해 리전마다 서비스의 엔드포인트/URL이 다를 수 있습니다. SDK에서 서비스에 사용할 엔드포인트를 결정할 수 있도록 클라이언트 객체를 구성할 때 리전을 제공합니다.

SDK

“SDK”라는 용어는 AWS SDK for PHP 라이브러리 전체를 가리킬 수도 있지만, 각 서비스의 클라이언트 객체를 위한 팩토리 역할을 하는 `Aws\Sdk` 클래스(문서)를 의미하기도 합니다. 또한 `Sdk` 클래스를 사용하여 생성되는 모든 클라이언트 객체에 적용되는 글로벌 구성 값 세트를 제공할 수 있습니다.

Service

AWS 서비스를 나타내는 일반적인 방법입니다(예: Amazon S3, Amazon DynamoDB, AWS OpsWorks 등). SDK에서 각 서비스에는 하나 이상의 API 버전을 지원하는 해당 클라이언트 객체가

있습니다. 또한 각 서비스에는 API를 구성하는 하나 이상의 작업이 있습니다. 서비스는 하나 이상의 리전에서 지원됩니다.

Signature

작업을 실행할 때 SDK에서는 보안 인증을 사용하여 요청에 대한 디지털 서명을 생성합니다. 그러면 서비스에서 요청을 처리하기 전에 서명을 확인합니다. 서명 프로세스는 SDK에 의해 캡슐화되며 클라이언트에 대해 구성된 보안 인증을 사용하여 자동으로 수행됩니다.

Waiter

Waiter는 리소스의 상태를 변경하고 일관적이거나 비동기적인 작업을 보다 쉽게 수행할 수 있도록 해주는 SDK의 기능입니다. 예를 들어, Amazon DynamoDB `CreateTable` 작업에서는 응답을 즉시 전송하지만 몇 초 동안 테이블이 액세스할 준비가 되지 않을 수 있습니다. Waiter를 실행하면 리소스의 상태를 절전 상태로 전환하고 폴링하여 리소스가 특정 상태로 전환될 때까지 대기할 수 있습니다. Waiter는 클라이언트의 `waitUntil()` 메서드를 사용하여 액세스합니다. 자세한 내용은 AWS SDK for PHP 버전 3 가이드를 참조하세요.

AWS 최신 용어는 AWS 일반 참조의 AWS 용어집을 참조하세요.

문서 기록

다음 표에서는 AWS SDK for PHP 개발자 가이드의 최신 릴리스 이후 변경된 중요 사항에 대해 설명합니다.

가장 최근 변경사항:

변경 사항	설명	날짜
목차 업데이트	다른 SDK 가이드와 더 일치하도록 목차를 수정했습니다.	2025년 7월 28일
체크섬을 통한 데이터 무결성 보호	자동 체크섬 계산에 대한 세부 정보로 콘텐츠가 업데이트되었습니다.	2025년 1월 16일
자격 증명 주제 개정	주제가 재구성되었습니다. 기본 자격 증명 공급자 체인에 대한 자세한 정보가 추가로 제공됩니다.	2025년 1월 10일
Amazon S3 암호화 클라이언트 V3 마이그레이션	V2에서 VAmazon S3V3 암호화 클라이언트 마이그레이션에 대한 주제 추가	2024년 12월 4일
Amazon S3 멀티파트 업로드	`ObjectUploader` 및 `MultipartUploader` 의 하위 명령을 구성하는 데 사용할 수 있는 'params' 배열을 설명합니다.	2024년 11월 6일
객체 업로더	S3 업로드 시 `ObjectUploader` 에서 사용 가능한 콜백의 사용을 명확하게 설명합니다.	2024년 10월 11일
Amazon S3 버킷 이름 업데이트	가이드 전체에서 S3 버킷 이름이 업데이트되었습니다.	2024년 9월 30일

Amazon EventBridge 글로벌 엔드포인트	Amazon EventBridge 글로벌 엔드포인트를 사용하는 방법을 보여주는 코드 예제 추가	2023년 12월 22일
AWS 공통 런타임(AWS CRT)	SDK for PHP의 AWS 공통 런타임(AWS CRT) 사용에 대해 설명하는 주제를 추가합니다.	2023년 11월 17일
StreamWrapper mkdir() 업데이트	`mkdir()`를 사용하여 버킷 및 폴더 객체 작업에 대한 정보를 추가합니다.	2023년 11월 2일
서비스 클라이언트 생성	'최신'이 기본값이므로 'version' 파라미터를 제거하여 코드 스니펫을 업데이트하세요.	2023년 8월 31일
기본 클라이언트 생성	'최신'이 기본값이므로 'version' 파라미터를 제거하여 코드 스니펫을 업데이트하세요.	2023년 8월 31일
새로운 목차	코드 예제에 더 쉽게 접근할 수 있도록 목차를 업데이트했습니다.	2023년 6월 1일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요. 시작하기 업데이트.	2023년 5월 20일
Amazon S3 멀티파트 업로드	동기 업로드를 위한 구성 정보가 포함되어 있습니다. 비동기 업로드를 위한 `add_content_md5` 업로드 옵션이 추가되었습니다.	2023년 4월 13일
Amazon S3 디렉터리 전송	`add_content_md5` 전송 옵션이 추가되었습니다.	2023년 4월 13일

참조 정보	AWS SDKs 및 도구 참조 안내서에 관련 세부 정보 콘텐츠에 대한 여러 링크가 추가되었습니다. 가이드 형식이 업데이트되었습니다.	2022년 9월 14일
일반 정리	AWS SDKs 및 도구 참조 가이드에 대한 참조가 추가되었습니다. 용어 업데이트를 반영하도록 AWS Key Management Service 섹션을 업데이트했습니다.	2022년 8월 23일
AWS 서비스 작업	GitHub에서 사용할 수 있는 코드 예제 목록이 포함됩니다.	2022년 4월 1일
SDK 지표 활성화	2021년 12월 20일에 더 이상 사용되지 않던 SDK 지표 활성화에 대한 정보가 삭제되었습니다.	2022년 1월 27일
Amazon S3 암호화 클라이언트 마이그레이션	Amazon S3 암호화 클라이언트 마이그레이션 주제 추가	2020년 8월 7일

이전 변경 사항:

변경	설명	릴리스 날짜
Secrets Manager 예제	다른 서비스 예제 추가	2019년 3월 27일
엔드포인트 검색	엔드포인트 검색 구성	2019년 2월 15일
Amazon CloudFront	다른 서비스 예제 추가	2019년 1월 25일
서비스 기능	SDK 지표	2018년 1월 11일
Amazon Kinesis, Amazon SNS	다른 서비스 예제 추가	2018년 12월 14일

변경	설명	릴리스 날짜
Amazon SES 예	다른 서비스 예제 추가	2018년 10월 5일
AWS KMS 예제	다른 서비스 예제 추가	2018년 8월 8일
자격 증명	보안 인증 가이드 설명 및 간소화	2018년 6월 30일
MediaConvert 예제	다른 서비스 예제 추가	2018년 6월 15일
새 웹 레이아웃	설명서가 AWS 스타일로 전환됨	2018년 9월 5일
Amazon S3 암호화	클라이언트 측 암호화	2017년 11월 17일
Amazon S3, Amazon SQS	다른 서비스 예제 추가	2017년 3월 26일
Amazon S3, IAM, Amazon EC2	다른 서비스 예제 추가	2017년 3월 17일
보안 인증 추가	AssumeRole 및 ini에 대한 지원 추가	2017년 1월 17일
S3 예제	S3 다중 리전 및 미리 서명된 게시물	2016년 3월 18일
OpenSearch Service 및 Amazon CloudSearch	다른 서비스 예제 추가	2015년 12월 28일
명령줄	명령 파라미터 추가	2015년 8월 13일
서비스 기능	S3 및에 대한 서비스 기능 추가 AWS	2015년 4월 30일
새 SDK 버전	의 버전 3이 AWS SDK for PHP 릴리스되었습니다.	2015년 26월 5일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.