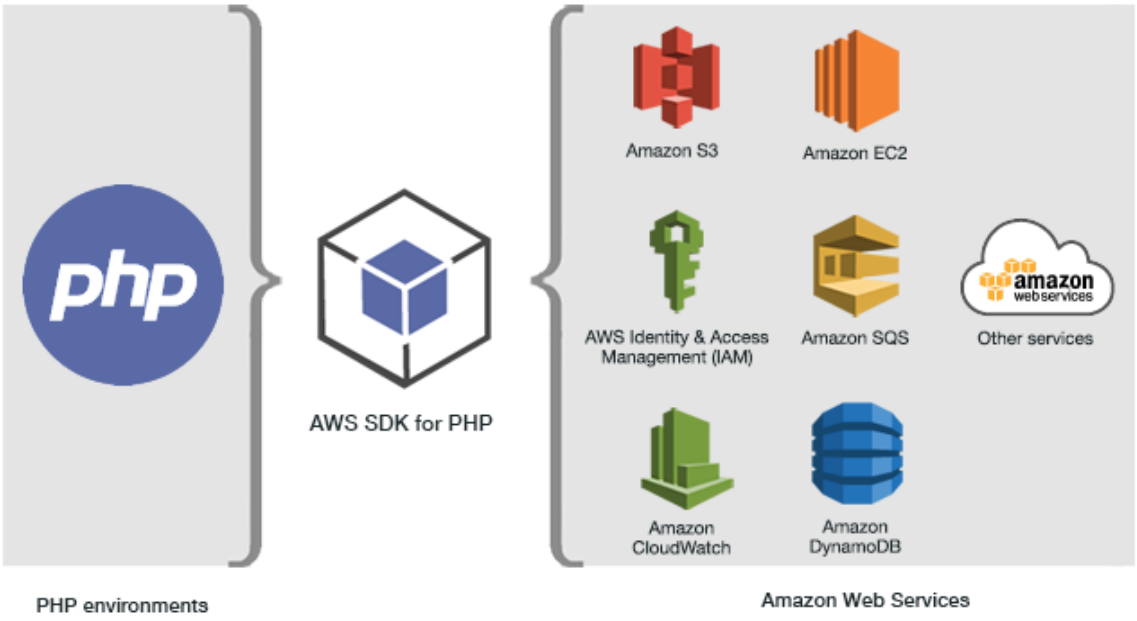
# PHP용 AWS SDK

## 개발자 가이드

# Table of Contents

# AWS SDK for PHP 버전 3은 무엇입니까?

PHP 개발자는 AWS SDK for PHP 버전 3으로 PHP 코드에 Amazon Web Services를 사용하고, Amazon S3, Amazon DynamoDB, Glacier 등의 서비스를 사용하여 강력한 애플리케이션과 소프트웨어를 빌드할 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. GitHub의 AWS SDK for PHP 버전 3에 대한 자세한 내용은 추가 리소스 (p. 276)를 참조하십시오.

외부 링크: API 문서 | GitHub | Gitter | 블로그 | 포럼 | Packagist

> **Note**
>
> SDK 버전 2를 사용하는 프로젝트 코드를 버전 3으로 마이그레이션하려면 PHP용 AWS SDK 버전 2에서 업그레이드 (p. 14)를 읽어 보십시오.

## 시작하기

- PHP용 AWS SDK 버전 3에 대한 요구 사항 및 권장 사항 (p. 4)
- PHP용 AWS SDK 버전 3 설치 (p. 5)
- PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)
- PHP용 AWS SDK 버전 2에서 업그레이드 (p. 14)

# SDK 가이드



# 서비스별 기능



# 예제



# 참조

- Guzzle 설명서

GitHub에 문제 제출:

- 설명서 문제 제출
- 버그 보고 또는 기능 요청

# API 설명서

SDK에 대한 API 설명서는 http://docs.aws.amazon.com/aws-sdk-php/v3/api/에서 확인할 수 있습니다.

# 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 PHP 버전 5.5.0 이상을 사용해야 합니다. 프라이빗 Amazon CloudWatch URL에 서명해야 하는 경우 OpenSSL PHP 확장도 필요합니다.

## 권장 사항

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 분석할 필요가 없습니다. 이 확장은 기본적으로 활성화됩니다.<br><br>Amazon Linux를 실행할 경우 OPCache 확장을 사용하려면 php56-opcache 또는 php55-opcache yum 패키지를 설치해야 합니다. |
| Xdebug 제거 | Xdebug를 사용하면 성능 병목 현상을 파악할 수 있습니다. 하지만 성능이 애플리케이션에 중요한 경우 Xdebug 확장을 프로덕션 환경에 설치하지 마십시오. 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.<br><br>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

SDK에서 `compatibility-test.php` 파일을 실행하여 시스템에서 SDK를 실행할 수 있는지 확인합니다. SDK의 최소 시스템 요구 사항을 충족하는 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 5.5 이상을 사용 중인지 확인하십시오.
환경의 요구 사항 및 권장 사항에 대해 자세히 알아보기 (p. 4)

## Composer를 통한 종속 항목으로 AWS SDK for PHP 설치

AWS SDK for PHP를 설치하려면 Composer를 사용하는 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 참조하십시오.

### Composer 설치

프로젝트에 Composer가 없는 경우, 다운로드하여 Composer를 설치하십시오.

Windows의 경우, Composer-Setup.exe를 다운로드 및 실행합니다.

Linux의 경우, Composer 다운로드 페이지의 명령줄 설치에 따르십시오.

## 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

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 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 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 3의 기본 사용 패턴

이 주제에서는 AWS SDK for PHP의 기본 사용 패턴을 중점적으로 살펴봅니다.

## 사전 조건

- SDK 다운로드 및 설치 (p. 5)
- AWS 액세스 키를 검색합니다.

## 코드에 SDK 포함

어떤 기술을 사용하여 SDK를 설치했든 상관없이, 단일 `require` 문을 사용하여 SDK를 코드에 포함시킬 수 있습니다. 설치 기법에 가장 적합한 PHP 코드를 찾으려면 다음 표를 참조하십시오. `/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.

## 사용법 요약

SDK를 사용하여 AWS 서비스와 상호 작용하려면 클라이언트 객체를 인스턴스화합니다. 클라이언트 객체에는 서비스 API의 작업과 일대일로 대응하는 메서드가 있습니다. 특정 작업을 실행하려면 해당 메서드를 호출합니다. 이 메서드는 성공 시 배열과 같은 결과 객체를 반환하고, 실패 시 예외를 발생시킵니다.

## 클라이언트 만들기

결합형 배열의 옵션을 클라이언트의 생성자에 전달하여 클라이언트를 만들 수 있습니다.

가져오기

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

use Aws\S3\S3Client;

use Aws\Exception\AwsException;
```

샘플 코드

```
//Create a S3Client
$s3 = new Aws\S3\S3Client([
    'profile' => 'default',
    'version' => 'latest',
```

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

자격 증명을 클라이언트에 명시적으로 제공하지 않았다는 점에 주의하십시오. 왜냐하면 SDK는 환경 변수 (p. 40)(`AWS_ACCESS_KEY_ID` 및 `AWS_SECRET_ACCESS_KEY` 이용), HOME 디렉터리의 AWS 자격 증명 INI 파일 (p. 41), AWS Identity and Access Management(IAM) 인스턴스 프로파일 자격 증명 (p. 42) 또는 자격 증명 공급자 (p. 46)로부터 자격 증명을 감지하기 때문입니다.

모든 일반 클라이언트 구성 옵션에 대해서는 구성 가이드 (p. 23)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.

가져오기

```
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',
    'version' => 'latest'
];

// 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. 서비스(예: "S3", "DynamoDb" 등)의 네임스페이스와 동일한 키에서 서비스별 구성 데이터를 제공할 수 있습니다.

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

# 서비스 작업 실행

클라이언트 객체에서 동일한 이름의 메서드를 호출하여 서비스 작업을 실행할 수 있습니다. 예를 들어, Amazon S3PutObject operation을 수행하려면 `Aws\S3\S3Client::putObject()` 메서드를 호출해야 합니다.

가져오기

```
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',
    'version' => 'latest'
];

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

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

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

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

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

클라이언트에 사용할 수 있는 작업과 입력 및 출력의 구조는 서비스 설명 파일을 기반으로 실행 시간에 정의됩니다. 클라이언트를 생성할 때 버전을 제공해야 합니다(예: "2006-03-01" 또는 "최신"). 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 옵션 (p. 30)을 사용하여 클라이언트를 인스턴스화할 때 설정할 수 있는 옵션과 동일합니다.

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

# 비동기 요청

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.

가져오기

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

샘플 코드

```
//Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => 'latest'
]);

// 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 마십시오. 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 a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => 'latest'
]);

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

# 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.

결합형 PHP 배열과 같이 AWSResult 객체의 데이터에 액세스할 수 있습니다.

가져오기

```
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',
    'version' => 'latest'
];

// 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 이 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');
```

# 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\S3Client`를 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.

가져오기

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

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

샘플 코드

```
//Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => 'latest'
]);

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

try {
    $s3Client->createBucket(['Bucket' => 'my-bucket']);
} catch (S3Exception $e) {
    // Catch an S3 specific exception.
    echo $e->getMessage();
} catch (AwsException $e) {
```

```
    // This catches the more generic AwsException. You can grab information
    // from the exception using methods of the exception object.
    echo $e->getAwsRequestId() . "\n";
    echo $e->getAwsErrorType() . "\n";
    echo $e->getAwsErrorCode() . "\n";
}
```

## 비동기 오류 처리

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

가져오기

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

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

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

가져오기

가져오기

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

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

샘플 코드

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

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

```
}
```

# AWS SDK for PHP 버전 2에서 업그레이드

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 새 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 페이지네이터 객체에서 데이터 쿼리 가능
- `'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 5.5 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 버전 3의 약속 (p. 58)을 참조하십시오.
- JMESPath의 PHP 구현(`mtdowling/jmespath.php`)은 SDK에서 `Aws\Result::search()` 및 `Aws\ResultPaginator::search()` 메서드의 데이터 쿼리 기능을 제공하는 데 사용됩니다. 자세한 내용은 PHP용 AWS SDK 버전 3의 JMESPath 표현식 (p. 77)을 참조하십시오.

## 이제 리전 및 버전 옵션이 필요합니다.

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 사항입니다. 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 인스턴스화에서 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 버전 3의 구성 (p. 23) 페이지를 참조하십시오.

Important

버전 3에서는 `'key'` 및 `'secret'`이 더 이상 루트 수준에서 유효한 옵션이 아니지만, `'credentials'` 옵션의 일부로 전달할 수 있습니다. 개발자가 AWS 자격 증명을 프로젝트에 하드코딩하지 않는 것이 좋기 때문입니다.

## SDK 객체

AWS SDK for PHP 버전 3에서는 `Aws\Common\Aws`를 대신하기 위해 `Aws\Sdk` 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 사용 (p. 8)에 설명되어 있습니다.

# 일부 API 결과가 변경됨

SDK가 API 작업 결과를 구문 분석하는 방법에 일관성을 기하기 위해 Amazon ElastiCache, Amazon RDS 및 Amazon Redshift 의 일부 API 응답에 래핑 요소를 추가했습니다.

예를 들어, 버전 3에서 Amazon RDSDescribeEngineDefaultParameters 결과를 호출하면 "EngineDefaults" 래핑 요소가 3개 포함되어 있습니다. 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 클래스 제거됨

AWS SDK for PHP 버전 2에 있던 `Enum` 클래스(예: `Aws\S3\Enum\CannedAcl`)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 클래스 제거

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 클래스 제거

AWS SDK for PHP 버전 2에는 `Aws` 클래스에서 `enableFacades()`를 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.

## 페이지네이터로 반복기 대체

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 페이지네이터 (p. 73)로 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 = $s3Client->getIterator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}
```

```
// Version 3
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($results as $result) {
    // You can extract any data that you want from the result.
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

페이지네이터 객체에는 JMESPath (p. 77) 표현식을 사용하여 결과 집합에서 데이터를 쉽게 추출할 수 있는 `search()` 메서드가 있습니다.

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

일반적으로 많은 상위 수준 추상화(서비스별 헬퍼 객체, 클라이언트와 별개로)가 개선되거나 업데이트되었습니다. 또한 일부는 제거되었습니다.

- Updated:
  - Amazon S3 멀티파트 업로드 (p. 204)를 사용하는 방법이 변경되었습니다. 멀티파트 업로드도 비슷한 방법으로 변경되었습니다.
  - Amazon S3에 미리 서명된 URL (p. 212)을 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 버전 3에서 Amazon S3 Transfer Manager 사용 (p. 100) 관련 설명서를 참조하십시오.
  - `Aws\S3\Model\ClearBucket` 및 `Aws\S3\Model\DeleteObjectsBatch`가 `Aws\S3\BatchDelete` 및 `S3Client::deleteMatchingObjects()`로 대체되었습니다.
  - PHP용 AWS SDK 버전 3에서 DynamoDB 세션 핸들러 사용 (p. 89) 옵션과 동작이 약간 변경되었습니다.
  - `Aws\DynamoDb\Model\BatchRequest` 네임스페이스가 `Aws\DynamoDb\WriteRequestBatch`로 대체되었습니다. DynamoDB WriteRequestBatch에 대한 설명서를 참조하십시오.
  - 이제 `SendRawEmail` 작업을 사용하면 `Aws\Ses\SesClient`가 `RawMessage`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 예에서는 AWS SDK for PHP 버전 3에서 버전 2와 달라진 몇 가지 사용 방법을 보여줍니다.

## 예제: 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' => 'my-bucket-name',
        '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' => 'my-bucket-name',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

## 예제: 전역 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.
```

# AWS SDK for PHP 버전 3 구성

AWS SDK for PHP는 다양한 기능과 구성 요소로 구성됩니다. 다음 각 항목에서는 SDK에서 사용되는 구성 요소를 설명합니다.

주제



# AWS SDK for PHP 버전 3 구성

이 가이드에서는 클라이언트 생성자 옵션에 대해 설명합니다. 이러한 옵션을 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.

구성 옵션

- version (p. 38)

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 기본 사용 설명서 (p. 7)를 참조하십시오.

# api_provider

형식

```
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

형식

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

특정 자격 증명 인스턴스에 사용할 `Aws\Credentials\CredentialsInterface` 객체를 전달합니다.

```
$credentials = new Aws\Credentials\Credentials('key', 'secret');

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

`credentials` 옵션을 제공하지 않으면 SDK는 다음과 같은 순서로 사용자의 환경에서 자격 증명을 로드하려고 시도합니다.

1. 환경 변수 (p. 40)에서 자격 증명을 로드합니다.
2. 자격 증명 .ini 파일 (p. 41)에서 자격 증명을 로드합니다.
3. IAM 역할 (p. 42)에서 자격 증명을 로드합니다.

null 자격 증명을 사용하고 요청에 서명하지 않으려면 `false`를 전달합니다.

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

```
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => false
]);
```

함수를 사용하여 자격 증명을 생성하려면 collable 자격 증명 공급자 (p. 46) 함수를 전달합니다.

```
use Aws\Credentials\CredentialProvider;

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

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

여러 프로세스 간의 기본 공급자 체인에서 반환된 값을 캐시하려면 `Aws\CacheInterface`의 인스턴스를 전달합니다.

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

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

AWD SDK for PHP 버전 3의 자격 증명 (p. 39) 가이드에서 자격 증명을 클라이언트에 제공하는 방법에 대한 자세한 내용을 참조할 수 있습니다.

> **Note**
>
> 자격 증명은 사용될 때 지연 로드되고 확인됩니다.

# 디버그

형식

```
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([
    'version' => 'latest',
    '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.

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

디버그 출력은 AWS SDK for PHP에서 문제를 진단할 때 매우 유용합니다. SDK에서 문제를 열 때 격리된 장애 사례에 대한 디버그 출력을 제공하십시오.

# stats

형식

```
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([
    'version' => 'latest',
    '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 활성화하려면 `true`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([
    'version' => 'latest',
    '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'];
```

# endpoint

형식

`string`

웹 서비스의 전체 URI입니다. 이 항목은 MediaConvert와 같이 계정별 엔드포인트를 사용하는 서비스에 필요합니다. 이러한 서비스의 경우 :doc`describeEndpoints`<emc-examples-getendpoint>` 메서드를 사용하여 이 엔드포인트를 요청합니다.

이 항목은 사용자 지정 엔드포인트에 연결할 때만 필요합니다(예: 로컬 버전의 Amazon S3 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 참조하십시오.

# endpoint_provider

형식

```
callable
```

"서비스" 및 "리전" 키를 포함한 옵션의 해시를 받는 선택적 PHP collable입니다. 이 항목은 `NULL` 또는 엔드포인트 데이터의 해시를 반환합니다. 여기서 "엔드포인트" 키는 필수입니다.

다음은 최소 엔드포인트 공급자를 생성하는 방법을 보여 주는 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

형식

```
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 비활성화하십시오. 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 사용하십시오.

```
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([
    'version' => 'latest',
    '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([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'endpoint_discovery' => new DoctrineCacheAdapter(new ApcuCache),
]);
```

배열을 엔드포인트 검색에 전달합니다.

```
use Aws\S3\S3Client;

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

# 핸들러

형식

```
    callable
```

명령 객체 및 요청 객체를 받아 `GuzzleHttp\Promise\PromiseInterface` 객체와 함께 실행되거나 `Aws\ResultInterface`과 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\MockHandler`를 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-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();
```

# http

형식

```
array
```

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

SDK는 다음과 같은 구성 옵션을 지원합니다.

## connect_timeout

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

```
use Aws\DynamoDb\DynamoDbClient;

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

## 디버그

형식

```
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

형식

```
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',
    'version' => 'latest',
    'http'    => ['decode_content' => false],
]);

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

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

## delay

형식

```
int
```

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

## expect

형식

```
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

형식

```
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',
    'version' => 'latest'
]);

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

## proxy

형식

```
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. 예: `"http://username:password@192.168.16.1:10"`.

- 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',
    'version' => 'latest',
    'http'    => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);

// Send requests through a different proxy per scheme
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'version' => 'latest',
    '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

형식

```
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

형식

```
bool
```

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

## stream

형식

```
bool
```

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

## timeout

형식

```
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',
    'version' => 'latest',
    'http'    => [
        'timeout' => 5
    ]
]);
```

## verify

형식

```
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',
    'version' => 'latest',
```

```
    'http'    => [
        'verify' => '/path/to/my/cert.pem'
    ]
]);

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

# http_handler

형식

```
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.

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

# profile

형식

```
string
```

HOME 디렉터리의 AWS 자격 증명 파일에서 자격 증명을 생성할 때 어떤 프로파일을 사용할지를 지정할 수 있습니다. 이 설정은 `AWS_PROFILE` 환경 변수를 재정의합니다.

> **Note**
>
> "프로파일"을 지정하면 "자격 증명" 키가 무시됩니다.

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

자격 증명 및 .ini 파일 형식 구성에 대한 자세한 내용은 AWD SDK for PHP 버전 3의 자격 증명 (p. 39)을 참조하십시오.

# region

형식

```
string
```

필수

```
true
```

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

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

# retries

형식

```
int
```

기본값

```
int(3)
```

클라이언트에 허용되는 최대 재시도 수를 구성합니다. 재시도를 비활성화하려면 0을 전달합니다.

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
]);
```

# 체계

형식

```
string
```

기본값

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

연결할 때 사용할 URI 체계입니다. 기본적으로 SDK는 "https" 엔드포인트를 사용합니다(SSL/TLS 연결 사용). scheme을 "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 `http` 체계를 지원하는지 여부는 AWS 리전 및 엔드포인트를 참조하십시오.

# service

형식

`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

형식

`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`을 반환하는 collable입니다. 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

형식

`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 예제에서는 서명 버전 4를 사용하도록 Amazon S3 클라이언트를 구성하는 방법을 보여 줍니다.

```
// 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_provider`는 사용자가 제공하는 `signature_version` 옵션을 생성할 수 있어야 합니다. SDK에서 사용되는 기본 `signature_provider`는 "v4" 및 "익명" 서명 버전에 대한 서명 객체를 생성할 수 있습니다.

# ua_append

형식

```
string|string[]
```

기본값

```
[]
```

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

# validate

형식

```
bool|array
```

기본값

```
bool(true)
```

클라이언트 측 파라미터 확인을 비활성화하려면 `false`로 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

형식

```
string
```

필수

```
true
```

사용할 웹 서비스의 버전입니다(예: `2006-03-01`).

"버전" 구성 값은 필수입니다. 버전 제약 조건을 지정하면 서비스가 갑작스럽게 변경되더라도 코드에 영향을 미치지 않습니다. 예를 들어, Amazon S3를 사용할 때 API 버전을 `2006-03-01`로 잠글 수 있습니다.

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

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.

클라이언트의 API 공급자에서 찾을 수 있는 가장 최신 API 버전을 사용하려면 `latest` 문자열을 "버전" 구성 값에 제공할 수 있습니다(기본 api_provider는 API용 SDK 모델의 `src/data` 디렉터리를 스캔함).

```
// Use the latest version available
$s3 = new Aws\S3\S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);
```

> **Warning**
>
> API 업데이트를 포함하는 SDK의 새 마이너 버전을 끌어오면 프로덕션 애플리케이션이 중단될 수 있으므로 프로덕션 애플리케이션에서 `latest`를 사용하지 않는 것이 좋습니다.

# AWS SDK for PHP 버전 3용 자격 증명

Amazon Web Services에 요청을 보내려면 자격 증명이라고 하는 AWS 액세스 키를 AWS SDK for PHP에 제공합니다.

다음과 같은 방법으로 추가가 가능합니다.

- 기본 자격 증명 공급자 체인(권장)을 사용합니다.
- 특정 자격 증명 공급자 또는 공급자 체인을 사용합니다(또는 직접 생성).
- 자체적으로 자격 증명을 제공합니다. 이러한 자격 증명은 루트 계정 자격 증명, IAM 자격 증명 또는 AWS STS에서 가져온 임시 자격 증명일 수 있습니다.

> **Important**
>
> 보안상 가급적이면 AWS 액세스용으로 루트 계정 대신에 IAM 사용자를 사용하는 것이 좋습니다. 자세한 내용은 IAM User Guide에서 IAM 모범 사례를 참조하십시오.

## 기본 자격 증명 공급자 체인 사용

자격 증명 인수를 제공하지 않고 새 서비스 클라이언트를 초기화하는 경우 SDK는 기본 자격 증명 공급자 체인을 사용하여 AWS 자격 증명을 찾습니다. SDK는 오류 없이 자격 증명을 반환하는 체인 내 첫 번째 자격 증명을 사용합니다.

기본 공급자 체인은 이 순서대로 다음과 같이 자격 증명을 찾아서 사용합니다.

1. 환경 변수의 자격 증명 사용 (p. 40).

   환경 변수 설정은 Amazon EC2 인스턴스 이외의 컴퓨터에서 개발 작업을 수행 중인 경우에 유용합니다.
2. AWS 공유 자격 증명 파일 및 프로파일 사용 (p. 41).

이 자격 증명 파일은 다른 SDK 및 AWS CLI에서 사용되는 것과 동일합니다. 공유 자격 증명 파일을 이미 사용 중인 경우 이 용도로 해당 파일을 사용할 수 있습니다.

PHP 코드 예제 중 대부분에서 이 메서드가 사용됩니다.

3. IAM 역할 수임 (p. 42).

   IAM 역할은 인스턴스의 애플리케이션에 AWS 호출용 임시 보안 자격 증명을 제공합니다. 예를 들면 IAM은 여러 Amazon EC2 인스턴스에서 자격 증명을 배포 및 관리할 수 있는 쉬운 방법을 제공합니다.

# 자격 증명을 추가하기 위한 그 밖의 방법

또한 다음과 같은 방식으로 자격 증명을 추가할 수도 있습니다.

- 자격 증명 공급자 사용 (p. 46).

  클라이언트 만들기 시 자격 증명용 사용자 지정 논리를 제공합니다.
- AWS STS의 임시 자격 증명 사용 (p. 51).

  2단계 인증 시 멀티 팩터 인증(MFA)을 사용하는 경우, AWS STS를 사용하여 AWS 서비스에 액세스하거나 AWS SDK for PHP를 사용할 수 있는 사용자 임시 자격 증명을 제공합니다.
- 하드 코딩된 자격 증명 사용 (p. 52)(권장되지 않음).

  Warning

  자격 증명을 하드 코딩하면 자격 증명을 SCM 리포지토리에 우연히 커밋할 수 있기 때문에 위험할 수 있습니다. 이렇게 하면 의도한 것보다 많은 사람에게 자격 증명이 노출될 수 있습니다. 또한 나중에 자격 증명을 교체하기 어려울 수 있습니다. 하드 코딩된 자격 증명을 포함하는 코드를 소스 제어에 제출하지 마십시오.

- 익명 클라이언트 만들기 (p. 53).

  서비스에서 익명 액세스를 허용하는 경우 어떤 자격 증명과도 연결되지 않은 클라이언트를 생성합니다.

자세한 내용은 Amazon Web Services 일반 참조의 AWS 보안 자격 증명 모범 사례를 참조하십시오.

# 환경 변수의 자격 증명 사용

환경 변수를 사용하여 자격 증명을 포함하면 AWS 보안 액세스 키가 실수로 공유되는 상황이 방지됩니다. 프로덕션 파일의 클라이언트에 AWS 액세스 키를 직접 추가하지 않는 것이 좋습니다. 누출된 키로 인해 계정이 손상된 개발자가 많습니다.

SDK는 Amazon Web Services에 인증하기 위해 먼저 환경 변수에서 자격 증명을 확인합니다. SDK는 `getenv()` 함수를 사용하여 `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` 및 `AWS_SESSION_TOKEN` 환경 변수를 찾습니다. 이러한 자격 증명을 환경 자격 증명이라고 합니다.

AWS Elastic Beanstalk에 애플리케이션을 호스팅하려는 경우 SDK에서 이러한 자격 증명을 자동으로 사용할 수 있도록 AWS Elastic Beanstalk 콘솔을 통해 `AWS_ACCESS_KEY_ID` 및 `AWS_SECRET_KEY` 환경 변수를 설정할 수 있습니다.

다음과 같이 명령줄에서 환경 변수를 설정할 수도 있습니다.

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your AWS account.
```

```
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS account.
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The session key for your AWS account. This is needed only when you are using temporary
 credentials.
   # 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 account.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your AWS account.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The session key for your AWS account. This is needed only when you are using temporary
 credentials.
   # 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 자격 증명 파일 및 자격 증명 프로필 사용

자격 증명은 액세스 키를 포함하는 일반 텍스트 파일입니다. 이 파일은 다음과 같아야 합니다.

- 애플리케이션을 실행 중인 동일 컴퓨터에 있어야 합니다.
- `credentials` 이름이 지정되어야 합니다.
- 홈 디렉터리의 `.aws/` 폴더에 있어야 합니다.

홈 디렉터리는 운영 체제마다 다를 수 있습니다. Windows에서는 환경 변수 `%UserProfile%`를 사용하여 홈 디렉터리를 참조할 수 있습니다. Unix-like 시스템에서는 환경 변수 `$HOME` 또는 `~`(물결표)를 사용할 수 있습니다.

다른 SDK 및 도구(예: AWS CLI)에 이 파일을 이미 사용하고 있는 경우 이 SDK에서 이 파일을 사용하기 위해 아무것도 할 필요가 없습니다. 다양한 도구나 애플리케이션에 대해 서로 다른 자격 증명을 사용하는 경우 프로파일을 사용하여 동일한 구성 파일에서 여러 액세스 키를 구성할 수 있습니다.

모든 PHP 코드 예제에서 이 메서드가 사용됩니다.

AWS 자격 증명 파일을 사용하면 다음과 같은 이점이 있습니다.

- 프로젝트의 자격 증명은 프로젝트 외부에 저장되므로

  실수로 자격 증명을 버전 관리로 커밋하는 상황은 발생하지 않습니다.
- 한 위치에서 여러 개의 자격 증명 세트를 정의하고 이름 지정할 수 있습니다.
- 동일한 자격 증명을 프로젝트 간에 쉽게 재사용할 수 있습니다.
- 그 밖의 AWS SDK 및 도구 지원, 이 동일한

  자격 증명 파일. 따라서 다른 도구에 자격 증명을 재사용할 수 있습니다.

AWS 자격 증명 파일의 형식은 다음과 유사합니다.

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

```
[project1]
aws_access_key_id = ANOTHER_AWS_ACCESS_KEY_ID
aws_secret_access_key = ANOTHER_AWS_SECRET_ACCESS_KEY
```

각 섹션(예: `[default]`, `[project1]`)은 별도의 자격 증명 프로파일을 나타냅니다. SDK 구성 파일에서, 또는 클라이언트를 인스턴스화하는 경우 `profile` 옵션을 사용하여 프로파일을 참조할 수 있습니다.

```
use Aws\DynamoDb\DynamoDbClient;

// Instantiate a client with the credentials from the project1 profile
$client = new DynamoDbClient([
    'profile' => 'project1',
    'region'  => 'us-west-2',
    'version' => 'latest'
]);
```

자격 증명 또는 프로파일을 SDK에 명시적으로 제공하지 않았고 환경 변수에서 자격 증명을 정의하지 않았지만, 자격 증명 파일을 정의한 경우 SDK는 "기본" 프로파일을 사용합니다. `AWS_PROFILE` 환경 변수에서 대체 프로파일 이름을 지정하여 기본 프로파일을 변경할 수 있습니다.

## 프로파일을 사용하여 역할 수임

`~/.aws/credentials`에서 역할에 대한 프로파일을 정의하여 IAM 역할을 사용하도록 PHP용 AWS SDK를 구성할 수 있습니다.

수임할 역할에 대해 `role_arn`을 사용하여 새 프로파일을 생성합니다. 또한 IAM 역할을 수임할 수 있는 권한을 가진 자격 증명을 사용하여 프로파일의 `source_profile`을 포함합니다.

다음 파일의 프로파일:
<problematic>`</problematic>
~/.aws/credentials':

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

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

클라이언트를 시작할 때 `AWS_PROFILE` 환경 변수 또는 `profile` 옵션을 설정하여 `default` 프로파일을 소스 자격 증명으로 사용하면 `project1`에 지정된 역할이 수임됩니다.

# IAM 역할 수임

## Amazon EC2 인스턴스 변수 자격 증명에 IAM 역할 사용

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

IAM 역할을 사용하는 경우 애플리케이션에서 자격 증명 관리에 대해 걱정할 필요가 없습니다. Amazon EC2 인스턴스의 메타데이터 서버에서 임시 자격 증명을 검색하여 인스턴스가 역할을 "수임"할 수 있습니다.

흔히 인스턴스 프로파일 자격 증명이라고 하는 이 임시 자격 증명을 사용하면 역할의 정책에 따라 허용되는 작업과 리소스에 액세스할 수 있습니다. Amazon EC2는 역할을 수임하기 위해 IAM 서비스에 대해 인스턴스를 안전하게 인증하고 검색한 역할 자격 증명을 주기적으로 새로 고치 등의 손이 많이 가는 작업을 모두 처리합니다. 이러한 방식으로 사용자의 별도 작업 없이 애플리케이션을 안전하게 보호할 수 있습니다.

Note

AWS Security Token Service(AWS STS)에서 생성된 인스턴스 프로파일 자격 증명 및 기타 임시 자격 증명은 일부 서비스에서 지원되지 않을 수 있습니다. 사용 중인 서비스가 임시 자격 증명을 지원하는지 여부를 확인하려면 AWS STS를 지원하는 AWS 서비스를 참조하십시오.
매번 메타데이터 서비스를 실행하지 않으려면 `Aws\CacheInterface`의 인스턴스를 `'credentials'` 옵션으로 클라이언트 생성자에 전달할 수 있습니다. 이렇게 하면 SDK가 캐시된 인스턴스 프로파일 자격 증명을 대신 사용할 수 있습니다. 자세한 내용은 PHP용 AWS SDK 버전 3 구성 (p. 23)을 참조하십시오.

## IAM 역할 생성 및 Amazon EC2 인스턴스에 할당

1. IAM 클라이언트를 생성합니다.

   가져오기

   ```
   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 클라이언트를 생성합니다.

   가져오기

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

```
use Aws\Ec2\Ec2Client;
```

샘플 코드

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

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

   샘플 코드

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

자세한 내용은 Amazon EC2의 IAM 역할을 참조하십시오.

## Amazon ECS 작업에 IAM 역할 사용

Amazon Elastic Container Service(Amazon ECS) 작업에 대한 IAM 역할을 사용하여 작업의 컨테이너에서 사용할 수 있는 IAM 역할을 지정할 수 있습니다. 이 기능은 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에서 자격 증명을 관리할 수 있는 전략을 제공합니다.

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

> **Note**
>
> AWS STS에서 생성된 인스턴스 프로파일 자격 증명 및 기타 임시 자격 증명은 일부 서비스에서 지원되지 않을 수 있습니다. 사용 중인 서비스가 임시 자격 증명을 지원하는지 여부를 확인하려면 AWS STS를 지원하는 AWS 서비스를 참조하십시오.

자세한 내용은 Amazon EC2 Container Service 작업의 IAM 역할을 참조하십시오.

## 다른 AWS 계정으로 IAM 역할 수임

AWS 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. 계정 간 액세스에 대한 자세한 내용은 자습서: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 단원을 참조하십시오.

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

AWS 계정에 대한 자격 증명을 사용하여 AWS STS 클라이언트를 생성합니다. 다음에서는 자격 증명 프로파일을 사용했지만 어떤 방법이든 사용할 수 있습니다. 새로 생성된 AWS STS 클라이언트를 사용하여 assume-role을 호출하고 사용자 지정 sessionName을 제공합니다. 결과에서 새 임시 자격 증명을 검색합니다. 기본적으로 자격 증명의 수명은 1시간입니다.

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 참조에서 IAM 역할 사용 또는 AssumeRole을 참조하십시오.

## 웹 자격 증명으로 IAM 역할 사용

웹 자격 증명 연동을 통해 고객이 AWS 리소스에 액세스할 때 인증하는 데 타사 자격 증명 공급자를 사용할 수 있습니다. 웹 자격 증명을 사용하여 역할을 수임하려면 먼저 IAM 역할을 생성하고 웹 자격 증명 공급자(IdP)를 구성해야 합니다. 자세한 내용은 웹 자격 증명 또는 OpenID Connect 연동을 위한 역할 생성(콘솔)을 참조하십시오.

자격 증명 공급자를 생성하고 웹 자격 증명을 위한 역할을 생성한 후 AWS STS 클라이언트를 사용하여 사용자를 인증합니다. 해당 사용자에 대한 권한으로 자격 증명에 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 참조에서 AssumeRoleWithWebIdentity - 웹 기반 자격 증명 공급자를 통한 연동 또는 AssumeRoleWithWebIdentity를 참조하십시오.

# 자격 증명 공급자 사용

자격 증명 공급자는 `GuzzleHttp\Promise\PromiseInterface` 인스턴스를 통해 이행되거나 `Aws\Credentials\CredentialsInterface`을 통해 거부되는 `Aws\Exception\CredentialsException`를 반환하는 함수입니다. 자격 증명 공급자를 사용하여 자격 증명 생성을 위한 고유의 사용자 지정 로직을 구현하거나 자격 증명 로딩을 최적화할 수 있습니다.

자격 증명 공급자는 `credentials` 클라이언트 생성자 옵션에 전달됩니다. 자격 증명 공급자는 비동기적이므로, API 작업을 호출할 때마다 강제로 늦게 평가됩니다. 따라서 자격 증명 공급자 함수를 SDK 클라이언트 생성자에 전달하면 자격 증명이 즉시 확인되지 않습니다. 자극 증명 공급자가 자격 증명 객체를 반환하지 않는 경우 API 작업은 `Aws\Exception\CredentialsException`을 통해 거부됩니다.

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

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

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

## SDK의 기본 제공 공급자

SDK는 사용자 지정 공급자와 함께 결합할 수 있는 여러 기본 제공 공급자를 제공합니다.

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

## assumeRole 공급자

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

> **Note**
>
> 모든 API 작업에서 AWS STS 자격 증명을 불필요하게 가져오지 않으려면 `memoize` 함수를 사용하여 만료될 때 자동으로 새로 고치는 자격 증명을 처리할 수 있습니다. 예제는 다음 코드를 참조하십시오.

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

// Passing Aws\Credentials\AssumeRoleCredentialProvider options directly
```

```
$assumeRoleCredentials = new AssumeRoleCredentialProvider([
    'client' => new StsClient([
        'region' => 'us-west-2',
        'version' => '2011-06-15'
    ]),
    'assume_role_params' => [
        'RoleArn' => 'arn:aws:iam::123456789012:role/role_name',
        'RoleSessionName' => 'test_session',
    ]
]);

// 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-west-2',
    'version'     => 'latest',
    'credentials' => $provider
]);
```

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

## 공급자 연결

`Aws\Credentials\CredentialProvider::chain()` 함수를 사용하여 자격 증명 공급자를 연결할 수 있습니다. 이 함수는 variadic 수의 인수를 받으며, 각 인수는 자격 증명 공급자 함수입니다. 그런 다음 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)
        )
    );
}
```

## 사용자 지정 공급자 생성

자격 증명 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 자격 증명 공급자를 생성하기 위해 호출되는 함수를 생성하는 것입니다. 예를 들어, 다음은 `env` 공급자의 소스입니다(예시용으로 약간 수정됨). 이 소스는 실제 공급자 함수를 반환하는 함수라는 점에 주의하십시오. 이 함수를 사용하여 자격 증명 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 && $secret) {
            return Promise\promise_for(
                new Credentials($key, $secret, getenv(self::ENV_SESSION))
            );
        }

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

## defaultProvider 공급자

`Aws\Credentials\CredentialProvider::defaultProvider`는 기본 자격 증명 공급자입니다. 이 공급자는 클라이언트를 생성할 때 `credentials` 옵션을 생략한 경우에 사용됩니다. 이 공급자는 먼저 환경 변수, 그 다음에 .ini 파일(`.aws/credentials` 파일을 먼저 시도하고, 다음에 `.aws/config` 파일), 그 다음에 인스턴스 프로파일(`EcsCredentials`를 먼저 시도하고, 다음에 `Ec2` 메타데이터)에서 자격 증명을 로드하려고 시도합니다.

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

## ecsCredentials 공급자

`Aws\Credentials\CredentialProvider::ecsCredentials`는 `GET` 요청을 통해 자격 증명을 로드하려고 시도합니다. 이 요청의 URI는 컨테이너에 있는 환경 변수 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`에서 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
]);
```

## env 공급자

`Aws\Credentials\CredentialProvider::env`는 환경 변수에서 자격 증명을 로드하려고 시도합니다.

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

$client = new S3Client([
    'region'      => 'us-west-2',
```

```
    'version'     => '2006-03-01',
    'credentials' => CredentialProvider::env()
]);
```

## ini 공급자

`Aws\Credentials\CredentialProvider::ini`는 ini 자격 증명 파일 (p. 41)에서 자격 증명을 로드하려고 시도합니다. 기본적으로 SDK는 `~/.aws/credentials`에 있는 파일에서 "기본" 프로파일을 로드하려고 시도합니다.

```
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
]);
```

## process 공급자

`Aws\Credentials\CredentialProvider::process`는 ini 자격 증명 파일 (p. 41)에서 지정하는 자격 증명 프로세스에서 자격 증명을 로드하려고 시도합니다. 기본적으로 SDK는 `~/.aws/credentials`에 있는 파일에서 "기본" 프로파일을 로드하려고 시도합니다. SDK가 credential_process 명령을 그대로 정확하게 호출한 후 stdout에서 JSON 데이터를 읽어옵니다. credential_process는 다음 형식으로 자격 증명을 stdout에 작성해야 합니다.

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

`SessionToken` 및 `Expiration`는 선택 사항입니다. 두 선택 사항을 지정한다면 자격 증명이 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
]);
```

## instanceProfile 공급자

`Aws\Credentials\CredentialProvider::instanceProfile`은 Amazon EC2 인스턴스 프로파일에서 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
]);
```

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

## 자격 증명 메모이제이션(memoization)

때로는 이전 반환 값을 기억하는 자격 증명 공급자를 생성해야 할 수도 있습니다. 이 공급자는 자격 증명 로딩이 비용이 많이 드는 작업인 경우 또는 `Aws\Sdk` 클래스를 사용하여 여러 클라이언트 간에 자격 증명 공급자를 공유하는 경우 성능에 유용합니다. 자격 증명 공급자 함수를 `memoize` 함수 안에 래핑하여 메모이제이션(memoization)을 자격 증명 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 자격 증명이 만료되면 메모이제이션 래퍼가 자격 증명을 새로 고치려고 시도할 때 래핑된 공급자를 호출합니다.

# AWS STS의 임시 자격 증명 사용

AWS Security Token Service(AWS STS)를 사용하여 AWS IAM 사용자 또는 자격 증명 연동을 통해 인증한 사용자에 대한 제한적 권한의 임시 자격 증명을 요청할 수 있습니다.

임시 자격 증명에 대한 일반 사용 사례 중 하나는 타사 자격 증명 공급자를 통해 사용자를 인증하여 모바일 또는 클라이언트 측 애플리케이션 액세스 권한을 AWS 리소스에 부여하는 것입니다(웹 자격 증명 연동 참조).

> Note
>
> AWS STS에서 생성된 임시 자격 증명이 모든 서비스에서 지원되는 것은 아닙니다. 사용 중인 서비스가 임시 자격 증명을 지원하는지 여부를 확인하려면 IAM 임시 보안 자격 증명을 참조하십시오.

## 임시 자격 증명 얻기

AWS STS에는 임시 자격 증명을 반환하는 여러 작업이 있지만, `GetSessionToken` 작업은 데모용으로 가장 간단한 작업입니다. `Aws\Sts\StsClient` 변수에 저장된 `$stsClient`의 인스턴스가 있다고 가정하면 이 인스턴스를 다음과 같이 호출합니다.

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

`GetSessionToken` 및 기타 AWS STS 작업의 결과에는 항상 `'Credentials'` 값이 포함됩니다. 결과를 인쇄하면(예: `print_r($result)`) 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 자격 증명 제공

클라이언트를 인스턴스화하고 AWS STS에서 직접 수신된 값을 전달하여 임시 자격 증명을 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 자격 증명을 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
]);
```

애플리케이션이나 프로젝트에서 임시 자격 증명 사용이 필요할 수 있는 이유에 대한 자세한 내용은 AWS STS 설명서의 임시 액세스 권한을 부여하기 위한 시나리오를 참조하십시오.

# 하드코딩 자격 증명 사용

새 서비스를 테스트하거나 문제를 디버깅할 경우 개발자는 클라이언트를 생성할 때 AWS 자격 증명을 포함시키려는 경우가 많습니다. AWS에 인증하는 방법에 대한 예는 다음 항목을 참조하십시오. 단, 주의를 기울이십시오. AWS SDK for PHP 버전 3용 자격 증명 (p. 39)에는 프로젝트에 자격 증명을 안전하게 추가하기 위한 다양한 권장 방법이 나와 있습니다.

> **Warning**
>
> 자격 증명을 하드 코딩하면 자격 증명을 SCM 리포지토리에 우연히 커밋할 수 있기 때문에 위험할 수 있습니다. 프로덕션 코드에 직접 인증 정보를 추가하면 의도한 것보다 더 많은 사용자에게 자격 증명이 노출될 수 있습니다. 또한 나중에 자격 증명을 교체하기 어려울 수 있습니다.

자격 증명을 SDK에 하드 코딩하기로 결정할 경우 "키", "암호" 및 선택적 "토큰" 키-값 페어의 결합형 배열을 클라이언트 생성자의 "자격 증명" 옵션에 제공하십시오.

```
// Hard-coded credentials
$s3Client = new S3Client([
    'version'     => 'latest',
    'region'      => 'us-west-2',
    'credentials' => [
        'key'    => 'my-access-key-id',
        'secret' => 'my-secret-access-key',
    ],
]);
```

## 익명 클라이언트 생성

자격 증명과 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' => 'my-bucket',
    'Key'    => 'my-key',
]);
```

# AWS SDK for PHP 버전 3의 명령 객체

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 상호작용할 필요는 없습니다. `Aws\S3\S3Client::putObject()`와 같은 메서드를 호출하면 SDK는 제공된 파라미터를 기반으로 `Aws\CommandInterface` 객체를 실제로 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' => 'foo',
    'Key'    => 'baz',
    'Body'   => 'bar'
];

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

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

# 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 옵션 (p. 30)을 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 옵션 (p. 36)과 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 `MockHandler`를 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\CommandPool`을 사용하면 `Aws\CommandInterface` 객체를 산출하는 반복자를 사용하여 명령을 동시에 실행할 수 있습니다. `CommandPool`은 풀의 명령을 반복하는 동안 일정한 수의 명령이 동시에 실행되도록 보장합니다(명령이 완료되면 일정한 풀 크기를 유지하기 위해 추가 명령이 실행됨).

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

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

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

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

이 예제는 `CommandPool`의 성능을 상당히 낮춘 것입니다. 더 복잡한 예제를 살펴보겠습니다. 디스크의 파일을 버킷에 업로드하려는 경우를 생각해 봅니다. 디스크에서 파일 목록을 가져오려면 PHP의 `DirectoryIterator`를 사용할 수 있습니다. 이 반복자는 `SplFileInfo` 객체를 생성합니다. `CommandPool`은 `Aws\CommandInterface` 객체를 산출하는 반복자를 받으므로, `Aws\CommandInterface` 객체를 반환하도록 `SplFileInfo` 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 = 'my-bucket';

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

// Create a generator that converts the SplFileInfo objects into
// Aws\CommandInterface objects. This generator accepts the iterator that
```

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

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

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

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

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

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

## CommandPool 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 3의 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, promise는 페이지네이터 (p. 74), 대기자 (p. 76), 명령 풀 (p. 56), 멀티파트 업로드 (p. 204), S3 디렉터리/버킷 전송 (p. 100) 등과 같이 SDK에서 제공하는 대부분의 상위 수준 추상화에서 사용됩니다.

`Async` 접미사가 있는 메서드를 호출하면 SDK가 제공하는 모든 클라이언트는 promise를 반환합니다. 예를 들어, 다음 코드는 Amazon DynamoDB`DescribeTable` 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 `describeTableAsync`를 호출할 수 있습니다. 이러한 메서드는 클라이언트와 연결된 API 모델 및 `__call` 번호로 구동되는 클라이언트의 magic `version` 메서드입니다. `describeTable` 접미사 없이 `Async`과 같은 메서드를 호출하면 클라이언트는 HTTP 요청을 전송하는 동안 차단하며 `Aws\ResultInterface` 객체를 반환하거나 `Aws\Exception\AwsException`을 발생시킵니다. 작업 이름에 `Async`(예: `describeTableAsync`) 접미사를 붙이면 클라이언트는 결국 `Aws\ResultInterface` 객체를 통해 이행되거나 `Aws\Exception\AwsException`을 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 `$onFulfilled` 및 `$onRejected`라는 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;

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

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

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

// Wait on both promises to complete and return the results
$results = Promise\unwrap($promises);

// Notice that this method will maintain the input array keys
```

```
var_dump($results['buckets']->toArray());
var_dump($results['tables']->toArray());
```

Note

CommandPool (p. 56)은 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`의 작업 대기열이 있는 경우 CommandPool (p. 56)을 사용합니다(명령은 메모리에 있거나 지연 반복자에서 산출될 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 생각해 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 = ['foo', 'baz', 'bar'];
$promises = [];

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

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

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

# AWS SDK for PHP 버전 3의 핸들러와 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
]);
```

`setHandler`의 `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);
```

## 모의(mock) 핸들러

SDK를 사용하는 테스트를 작성할 경우 `MockHandler`를 사용하는 것이 좋습니다. `Aws\MockHandler`를 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 명

령에 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');
```

validate

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');
```

build

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 `appendBuild`은 `build` 목록의 끝에 미들웨어를 추가하고, `prependBuild`은 `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 `HandlerList`의 마지막 단계입니다.

`sign` 및 `appendSign` 메서드를 사용하여 `prependSign` 단계에 미들웨어를 추가할 수 있습니다. 여기서 `appendSign`은 `sign` 목록의 끝에 미들웨어를 추가하고, `prependSign`은 `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 callable을 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'] = 'mybucket';
        // 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' => 'mybucket'
]);

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

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' => 'mybucket'
]);

$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.

## history

`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 callable 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 (p. 30)
- 디버그 (p. 30)
- decode_content (p. 31)(선택 사항)
- delay (p. 31)
- progress (p. 32)(선택 사항)
- proxy (p. 32)
- sink (p. 33)
- synchronous (p. 33)(선택 사항)
- stream (p. 34)(선택 사항)
- timeout (p. 34)
- verify (p. 34)
- http_stats_receiver(선택 사항) - stats (p. 26) 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 이외에 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 Streams

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 BLOB로 정의된 모든 명령은 문자열, PHP 스트림 리소스 또는 `Psr\Http\Message\StreamInterface` 인스턴스로 만족시킬 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 3의 페이지네이터

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

```
$results = $s3Client->getPaginator('ListObjects', [
    'Bucket' => 'my-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 `foreach`와 함께 작동합니다. 또한 이 객체는 `iterator_to_array`와 마찬가지로 반복자 함수와 함께 사용할 수 있으며, `LimitIterator` 객체와 마찬가지로 SPL 반복자와 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 S3 `ListObjects` 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 표현식 (p. 77)을 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' => 'my-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'    => 'my-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' => 'my-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 결과입니다. 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' => 'my-bucket',
            'Key'    => 'Foo'
        ]);
    }
});

$promise
    ->then(function ($result) {
        // Result would be the last result to the deleteAsync operation
```

```
    })
    ->otherwise($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 Waiters

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.

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

```
// Create a bucket
$s3Client->createBucket(['Bucket' => 'my-bucket']);

// Wait until the created bucket is available
$s3Client->waitUntil('BucketExists', ['Bucket' => 'my-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'  => 'my-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'  => 'my-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' => 'my-bucket'];

// Create a waiter promise
$waiter = $s3Client->getWaiter($waiterName, $waiterOptions);

// Initiate the waiter and retrieve a promise
$promise = $waiter->promise();

// Call methods when the promise is resolved.
$promise
    ->then(function () {
        echo "Waiter completed\n";
    })
    ->otherwise(function (\Exception $e) {
        echo "Waiter failed: " . $e . "\n";
    });

// Block until the waiter completes or fails. Note that this might throw
// a \RuntimeException if the waiter fails.
$promise->wait();
```

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 페이지 네이터 (p. 73), AWS SDK for PHP 버전 3의 Waiter (p. 75))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 용으로 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 아래와 같은 기본 JSON 출력으로 시작합니다. 여기서는 별도의 Amazon 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 3 (p. 73) 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 Version 3의 SDK 지표

Enterprise Support에 대한 AWS SDK 지표(SDK 지표)는 엔터프라이즈 고객이 AWS Enterprise Support와 공유하고 있는 자신의 호스트 및 클라이언트에서 AWS SDK 지표를 수집할 수 있는 옵션입니다. SDK 지표는 AWS Enterprise Support 고객을 위해 AWS 서비스에 연결 중에 발생하는 문제의 감지 및 진단 속도를 높이는 데 도움이 되는 정보를 제공합니다.

원격 측정은 각 호스트에서 수집되므로 UDP를 통해 로컬 호스트로 릴레이됩니다. 여기서 CloudWatch 에이전트는 데이터를 집계하여 SDK 지표 서비스로 전송합니다. 따라서 지표를 받으려면 CloudWatch 에이전트를 인스턴스에 추가해야 합니다.

SDK 지표에 대한 자세한 내용은 Amazon CloudWatch User Guide를 참조하십시오.

다음 각 주제는 AWS SDK for PHP에서 SDK 지표를 설정하고, 구성하고, 관리하는 방법을 설명하고 있습니다.

주제



## AWS SDK for PHP 버전 3에서 지표를 수집하여 전송할 수 있는 권한을 SDK 지표에게 부여

고객이 Enterprise Support에 대한 AWS SDK 지표 Enterprise를 사용해 AWS SDK에서 지표를 수집하려면 CloudWatch 에이전트 권한을 부여하는 IAM 역할을 생성하여 Amazon EC2 인스턴스 또는 프로덕션 환경에서 데이터를 수집해야 합니다.

다음 PHP 코드 샘플 또는 AWS 콘솔을 사용해 CloudWatch 에이전트가 고객 환경에서 SDK 지표에 액세스할 때 필요한 IAM 정책과 역할을 생성합니다.

AWS SDK for PHP에서 SDK 지표를 사용하는 방법에 대한 자세한 내용은 PHP용 AWS SDK 버전 3에서 SDK 지표 설정 (p. 83) 단원을 참조하십시오. SDK 지표에 대한 자세한 내용은 Amazon CloudWatch User Guide에서 SDK 지표에 대한 IAM 권한 단원을 참조하십시오.

### AWS SDK for PHP를 사용하여 액세스 권한 설정

Amazon EC2 Systems Manager 및 SDK 지표에 대한 권한이 부여된 인스턴스에게 IAM 역할을 생성합니다.

먼저 CreatePolicy를 사용해 정책을 생성합니다. 그런 다음 CreateRole을 사용해 역할을 생성합니다. 마지막으로 AttachRolePolicy를 사용해 생성된 정책을 새로운 역할에 연결합니다.

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new IamClient([
```

```
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-05-08'
]);

$roleName = 'AmazonCSM';

$description = 'An Instance role that has permission for  Amazon EC2 Systems Manager  and
 SDK Metric Monitoring.';

$AmazonCSMPolicy = '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "sdkmetrics-beta:*"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ssm:GetParameter"
                ],
                "Resource": "arn:aws:ssm:*:*:parameter/AmazonCSM*"
            }
        ]
    }';

$rolePolicy = '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}';


try {
    $iamPolicy = $client->createPolicy([
        'PolicyName' => $roleName . 'policy',
        'PolicyDocument' => $AmazonCSMPolicy
    ]);
    if ($iamPolicy['@metadata']['statusCode'] == 200) {
        $policyArn = $iamPolicy['Policy']['Arn'];
        echo('<p> Your IAM Policy has been created. Arn -  ');
        echo($policyArn);
        echo('<p>');
        $role = $client->createRole([
            'RoleName' => $roleName,
            'Description' => $description,
            'AssumeRolePolicyDocument' => $rolePolicy,
        ]);
        echo('<p> Your IAM User Role has been created. Arn: ');
        echo($role['Role']['Arn']);
        echo('<p>');
        if ($role['@metadata']['statusCode'] == 200) {
            $result = $client->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
```

```
            ]);
            var_dump($result)
        } else {
            echo('<p> There was an error creating your IAM User Role </p>');
            var_dump($role);
        }
    } else {
        echo('<p> There was an error creating your IAM Policy </p>');
        var_dump($iamPolicy);

    }
} catch (AwsException $e) {
    // output error message if fails
    echo $e;
    error_log($e->getMessage());

}
```

## IAM 콘솔을 사용하여 액세스 권한 설정

또는 IAM 콘솔을 사용하여 역할을 생성할 수 있습니다.

1. IAM 콘솔로 이동하여 Amazon EC2를 사용할 역할을 생성합니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. AWS 서비스와 EC2를 차례대로 선택합니다.
5. Next: Permissions(다음: 권한)을 선택합니다.
6. Attach permissions policies(권한 정책 연결) 아래에서정책 생성을 선택합니다.
7. 서비스에서 Systems Manager(시스템 관리자)를 선택합니다. 작업에서 `Read`를 확장한 후 `GetParameters`를 선택합니다. 리소스에 대하여 CloudWatch 에이전트를 지정합니다.
8. 권한 추가
9. Choose a service(서비스 선택)를 선택한 다음 Enter service manually(수동으로 서비스 입력)를 선택합니다. 서비스에 `sdkmetrics`를 입력합니다. 모든 `sdkmetrics` 작업과 모든 리소스를 선택한 후 정책 검토를 선택합니다.
10. 역할 `AmazonSDKMetrics`의 이름을 지정하고 설명을 추가합니다.
11. 역할 생성을 선택합니다.

# AWS SDK for PHP 버전 3에서 SDK 지표 설정

다음 단계에서는 AWS SDK for PHP에 대하여 SDK 지표을 설정하는 방법을 보여줍니다. 이러한 단계는 AWS SDK for PHP를 사용하는 클라이언트 애플리케이션에 대해 Amazon Linux를 실행하는 Amazon EC2 인스턴스와 관련이 있습니다. SDK 지표은 AWS SDK for PHP를 구성하는 동안 활성화한 경우 프로덕션 환경에서 사용할 수도 있습니다.

SDK 지표을 활용하려면 CloudWatch 에이전트의 최신 버전을 실행합니다. Amazon CloudWatch User Guide에서 SDK 지표에 대한 CloudWatch 에이전트 구성 방법을 알아봅니다.

SDK 지표에 대한 IAM 권한에 대한 자세한 내용은 PHP용 AWS SDK 사용 시 SDK 지표에 대한 IAM 권한 (p. 81) 단원을 참조하십시오.

AWS SDK for PHP를 사용하여 SDK 지표를 설정하려면 다음 지침을 따릅니다.

1. AWS SDK for PHP 클라이언트로 AWS 서비스를 사용할 애플리케이션을 생성합니다.
2. Amazon EC2 인스턴스 또는 로컬 환경에서 프로젝트를 호스팅합니다.
3. AWS SDK for PHP의 최신 버전을 설치하여 사용합니다.
4. EC2 인스턴스 또는 로컬 환경에 CloudWatch 에이전트를 설치하고 구성합니다.
5. 지표를 수집 및 전송하도록 SDK 지표를 승인합니다.
6. PHP용 AWS SDK 사용 시 SDK 지표 활성화 (p. 84).

자세한 정보는 다음을 참조하십시오.

- CloudWatch 에이전트 업데이트 (p. 85)
- SDK 지표 비활성화 (p. 85)

# AWS SDK for PHP 사용 시 SDK 지표 활성화

기본적으로 SDK 지표는 꺼져 있으며 포트는 31000으로 설정되어 있습니다. 다음은 기본 파라미터입니다.

```
//default values
 [
     'enabled' => false,
     'port' => 31000,
 ]
```

SDK 지표 활성화는 AWS 서비스를 사용하도록 자격 증명을 구성하는 작업과 관련이 없습니다.

환경 변수를 설정하거나 AWS 공유 구성 파일을 사용하면 SDK 지표를 활성화할 수 있습니다.

## 옵션 1: 환경 변수 설정

SDK는 먼저 `AWS_PROFILE` 아래의 환경 변수에 지정된 프로필에서 SDK 지표이 활성화되어 있는지 확인합니다.

SDK 지표를 켜려면 다음을 환경 변수에 추가합니다.

```
export AWS_CSM_ENABLED=true
```

기타 구성 설정 (p. 85)을 사용할 수 있습니다. 공유 파일 사용에 대한 자세한 내용은 환경 변수에서 자격 증명 사용 (p. 40) 단원을 참조하십시오.

> **Note**
>
> SDK 지표을 활성화하면 AWS 서비스를 사용하도록 자격 증명이 구성되지 않습니다. 이를 위해서는 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39) 단원을 참조하십시오.

## 옵션 2: AWS 공유 구성 파일

CSM 구성이 환경 변수에서 발견되지 않으면 SDK는 기본 AWS 프로필 필드를 찾습니다. `AWS_DEFAULT_PROFILE`이 기본값이 아닌 다른 값으로 설정된 경우 해당 프로필을 업데이트합니다. SDK 지표를 활성화하려면 `csm_enabled`를 `~/.aws/config`에 위치한 공유 구성 파일에 추가합니다.

```
[default]
csm_enabled = true

[profile aws_csm]
csm_enabled = true
```

기타 구성 설정 (p. 85)을 사용할 수 있습니다. AWS 공유 파일 사용에 대한 자세한 내용은 AWS 자격 증명 파일 및 자격 증명 프로필 사용하기 (p. 41) 단원을 참조하십시오.

Note

SDK 지표을 활성화하면 AWS 서비스를 사용하도록 자격 증명이 구성되지 않습니다. 이를 위해서는 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39) 단원을 참조하십시오.

# CloudWatch 에이전트 업데이트

포트 ID를 변경하려면 값을 설정한 다음 현재 활성 상태인 모든 AWS 작업을 다시 시작해야 합니다.

## 옵션 1: 환경 변수 설정

대부분의 AWS 서비스는 기본 포트를 사용합니다. 하지만 SDK 지표에서 모니터링할 서비스가 고유 포트를 사용하는 경우, AWS_CSM_PORT=[port_number]를 호스트의 환경 변수에 추가합니다.

```
export AWS_CSM_ENABLED=true
export AWS_CSM_PORT=1234
```

## 옵션 2: AWS 공유 구성 파일

대부분의 서비스는 기본 포트를 사용합니다. 그러나 서비스에 고유한 포트 ID가 필요한 경우에는 csm_port = [port_number]를 ~/.aws/config에 추가합니다.

```
[default]
csm_enabled = false
csm_port = 1234

[profile aws_csm]
csm_enabled = false
csm_port = 1234
```

## SDK 지표 다시 시작

작업을 다시 시작하려면 다음 명령을 실행합니다.

```
amazon-cloudwatch-agent-ctl -a stop;
amazon-cloudwatch-agent-ctl -a start;
```

# SDK 지표 비활성화

SDK 지표를 끄려면 환경 변수 또는 `~/.aws/config`에 있는 AWS 공유 구성 파일에서 csm_enabled를 false로 설정합니다. 그런 다음 변경 사항이 적용될 수 있도록 CloudWatch 에이전트를 다시 시작합니다.

## csm_enabled를 false로 설정

옵션 1: 환경 변수

```
export AWS_CSM_ENABLED=false
```

옵션 2: AWS 공유 구성 파일

Note

환경 변수는 AWS 공유 구성 파일을 재정의합니다. SDK 지표가 환경 변수에서 활성화되어 있으면 SDK 지표은 활성화된 상태로 유지됩니다.

```
[default]
csm_enabled = false

[profile aws_csm]
csm_enabled = false
```

### SDK 지표 중단 후 CloudWatch 에이전트 재시작

SDK 지표을 비활성화하려면 다음 명령을 사용합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a stop &&
echo "Done"
```

다른 CloudWatch 기능을 사용하는 경우 다음 명령을 사용하여 CloudWatch을 다시 시작합니다.

```
amazon-cloudwatch-agent-ctl -a start;
```

## SDK 지표 용어 정의

SDK 지표에 대한 다음 설명을 사용하여 결과를 해석할 수 있습니다. 일반적으로 이러한 지표는 정기적인 비즈니스 검토 중에 기술 지원 관리자와 검토할 수 있습니다. AWS Support 리소스 및 기술 지원 관리자는 사례를 해결할 수 있도록 SDK 지표 데이터에 대한 액세스 권한이 있어야 합니다. 그러나 혼란을 주거나 예기치 않은 데이터를 발견했지만 애플리케이션의 성능에 부정적인 영향을 미치지 않는 경우에는 예약된 비즈니스 검토 중에 해당 데이터를 검토하는 것이 좋습니다.

| 지표: | CallCount |
| --- | --- |
| 정의 | 해당 코드에서 AWS 서비스에 대해 수행한 호출 중 성공하거나 실패한 총 API 호출 수 |
| 사용 방법 | 이 총 호출 수를 기준으로 사용하여 오류 또는 조절과 같은 기타 지표와의 상관관계를 보여줍니다. |

| 지표: | ClientErrorCount |
| --- | --- |
| 정의 | 클라이언트 오류(4xx HTTP 응답 코드)와 함께 실패한 API 호출 수. 예: 조절, 거부된 액세스, S3 버킷 없음, 잘못된 파라미터 값. |
| 사용 방법 | 조절과 관련된 특정 사례를 제외하고(예: 증가시킬 필요가 있는 제한에 따라 조절이 발생하는 경우) 이 지표는 고정될 필요가 있는 애플리케이션 내의 무언가를 지시할 수 있습니다. |

| 지표: | ConnectionErrorCount |
| --- | --- |
| 정의 | 서비스 연결 오류로 인해 실패한 API 호출 수. 이는 로드 밸런서, DNS 오류, 전송 공급자를 포함하여 고객 애플리케이션과 AWS 서비스 간의 네트워크 문제로 인해 발생할 수 있습니다. 경우에 따라 AWS 문제로 인해 이 오류가 발생할 수 있습니다. |

| 지표: | ConnectionErrorCount |
| --- | --- |
| 사용 방법 | 이 지표를 사용하여 문제가 애플리케이션에 한정된 문제인지 아니면 인프라 및/또는 네트워크에 의해 발생하는지 확인합니다. 높은 ConnectionErrorCount 값은 API 호출에 대한 짧은 시간 초과 값을 나타낼 수도 있습니다. |

| 지표: | ThrottleCount |
| --- | --- |
| 정의 | AWS 서비스의 조절 때문에 실패한 API 호출 수. |
| 사용 방법 | 이 지표를 사용하여 애플리케이션이 조정(throttling) 한계에 도달했는지 평가하고 재시도 및 애플리케이션 대기 시간의 원인을 확인할 수 있습니다. 호출을 배치(batch)로 처리하는 대신 기간 동안 호출을 분산시켜 보십시오. |

| 지표: | ServerErrorCount |
| --- | --- |
| 정의 | AWS 서비스의 서버 오류(5xx HTTP 응답 코드)로 인해 실패한 API 호출 수. 이 실패의 원인은 일반적으로 AWS 서비스입니다. |
| 사용 방법 | SDK 재시도 또는 지연 시간의 원인을 결정합니다. 일부 AWS 팀에서는 지연 시간을 HTTP 503 응답으로 분류하기 때문에 이 지표가 항상 AWS 서비스에 결함이 있음을 나타내는 것은 아닙니다. |

| 지표: | EndToEndLatency |
| --- | --- |
| 정의 | 애플리케이션이 AWS SDK를 사용하여 호출(재시도 포함)을 수행한 총 시간. 즉, 성공 여부와 상관없이 여러 번 시도한 후 또는 재시도가 불가능한 오류에 따라 호출이 실패하는 즉시 호출을 수행한 총 시간. |
| 사용 방법 | AWS API 호출이 애플리케이션의 전체 지연 시간에 얼마나 기여했는지 확인합니다. 예상보다 높은 지연 시간은 네트워크, 방화벽이나 기타 구성 설정 문제 또는 SDK 재시도의 결과로 발생하는 지연 시간으로 인해 발생할 수 있습니다. |

# AWS SDK for PHP 버전 3을 통한 AWS Services 사용

AWS SDK for PHP 버전 3에서 지원되는 일부 AWS 서비스에는 API에 대한 작업 실행 이외의 추가 기능이 포함되어 있습니다. 이 가이드에서는 이러한 서비스별 상위 수준 기능에 대해 설명합니다.

주제



## AWS SDK for PHP 버전 3에서 AWS Cloud9 사용

AWS SDK for PHP 버전 3에서 AWS Cloud9를 사용하면 브라우저만 있어도 PHP 코드를 쓰고, 실행하고, 디버깅할 수 있습니다. AWS Cloud9에는 코드 편집기, 디버거, 터미널 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 참조하십시오.

다음 지침에 따라 AWS SDK for PHP에서 AWS Cloud9를 설정하십시오.



### 1단계: AWS Cloud9을 사용하도록 AWS 계정 설정

AWS Cloud9에 대한 액세스 권한이 있는 AWS 계정에서 AWS Identity and Access Management(IAM) 개체(예: IAM 사용자)로 AWS Cloud9에 로그인하여 AWS Cloud9 사용을 시작합니다.

AWS Cloud9에 액세스하고 AWS Cloud9 콘솔에 로그인할 수 있도록 AWS 계정에 IAM 개체를 설정하는 방법은 AWS Cloud9 사용 설명서의 AWS Cloud9 팀 설정을 참조하십시오.

### 2단계: AWS Cloud9 개발 환경 설정

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

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

> 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 를 사용하는 가장 빠른 방법입니다.

## 3단계: AWS SDK for PHP 설정

AWS Cloud9에서 개발 환경용 IDE가 열리면 다음과 같이 IDE를 사용하여 해당 환경에서 AWS SDK for PHP를 설정합니다.

1. 터미널이 IDE에 아직 열려 있지 않은 경우 터미널을 엽니다. IDE의 메뉴 모음에서 Window, New Terminal(창, 새 터미널)을 선택합니다.
2. 다음 명령을 실행하여 AWS SDK for PHP를 설치합니다.

```
curl -sS https://getcomposer.org/installer | php
php composer.phar require aws/aws-sdk-php
```

IDE에서 PHP를 찾을 수 없는 경우 다음 명령을 실행하여 PHP를 설치합니다. 이 명령은 이 항목의 앞 부분에서 Create a new instance for environment (EC2)(환경에 대한 새 인스턴스 생성(EC2)) 옵션을 선택한 것으로 가정합니다.

```
sudo yum -y install php56
```

## 4단계: 예제 코드 다운로드

이전 단계에서 연 터미널을 사용하여 AWS SDK for PHP에 대한 예제 코드를 AWS Cloud9 개발 환경으로 다운로드합니다.

이를 위해 다음 명령을 실행합니다. 이 명령은 공식 AWS SDK 설명서에 사용되는 모든 코드 예제를 환경의 루트 디렉터리로 다운로드합니다.

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

AWS SDK for PHP에 대한 코드 예제를 찾으려면 환경 창을 사용하여 `ENVIRONMENT_NAMEaws-doc-sdk-examplesphpexample_code` 디렉터리를 엽니다. 여기서 `ENVIRONMENT_NAME`은 개발 환경의 이름입니다.

이러한 예제와 기타 코드 예제를 작동하는 방법에 대한 자세한 내용은 코드 예제 (p. 107)를 참조하십시오.

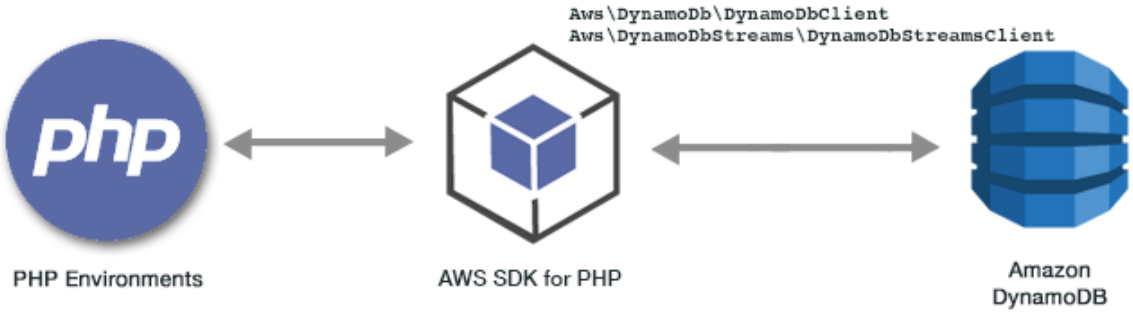
## 5단계: 예제 코드 실행 및 디버깅

AWS Cloud9 개발 환경에서 코드를 실행하는 방법은 AWS Cloud9 사용 설명서의 코드 실행을 참조하십시오.

코드를 디버깅하는 방법은 AWS Cloud9 사용 설명서의 코드 디버깅을 참조하십시오.

# AWS SDK for PHP 버전 3을 통해 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 설치하기 쉽고 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 참조하십시오.

# 기본 사용법

## 1단계: 핸들러 등록

먼저 세션 핸들러를 인스턴스화하고 등록합니다.

```
use Aws\DynamoDb\SessionHandler;

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

$sessionHandler->register();
```

## 단계 2. 세션을 저장할 테이블 생성

실제로 세션 핸들러를 사용하기 전에 먼저 세션을 저장할 테이블을 생성해야 합니다. Amazon DynamoDB용 AWS 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.

## 단계 3. 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 DynamoDB를 사용하여 반복을 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. 이 옵션의 기본값은 `'sessions'`입니다.

**`hash_key`**

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

**`data_attribute`**

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

**`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 `10000`이며 세션 잠금에만 사용됩니다.

**`max_lock_retry_microtime`**

세션 핸들러가 잠금을 획득하기 위한 시도 간에 대기해야 하는 최대 시간(마이크로초)입니다. 기본값은 `50000`이며 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',
    '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,
]);
```

# 요금

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 홈페이지에 따르면 다음과 같습니다.

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(`consistent_read`가 `false`인 경우 0.5만).<br>• (조건부) 세션이 만료된 경우 세션을 삭제하기 위한 1개의 쓰기 작업입니다. |
| `session_start()`를 통한 읽기(세션 잠금 사용) | • 최소 1개의 쓰기 작업입니다.<br>• (조건부) 세션에서 잠금 획득 시 각 시도에 대해 추가 쓰기 작업입니다. 구성된 잠금 대기 시간 및 재시도 옵션을 기반으로 합니다.<br>• (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.<br>• 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 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 고려하십시오.

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.

이러한 이유로 AWS는 자동화된 가비지 수집을 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 리전에서 세션을 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 사용되는 처리량을 모니터링하고 필요에 따라 애플리케이션의 수요에 맞게 처리량을 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 마십시오.
6. PHP의 내장 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.

# 필요한 IAM 권한

DynamoDB SessionHhandler를 사용하려면 구성된 자격 증명 (p. 39)에 이전 단계에서 생성 (p. 90)한 DynamoDB 테이블을 사용할 수 있는 권한이 있어야 합니다. 다음 IAM 정책에는 필요한 최소 권한이 포함되

어 있습니다. 이 정책을 사용하려면 리소스 값을 이전에 생성한 테이블의 Amazon Resource Name(ARN)으로 바꿉니다. IAM 정책을 만들고 연결하는 방법에 대한 자세한 내용은 IAM User Guide의 IAM 정책 관리를 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:Scan",
        "dynamodb:BatchWriteItem"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:dynamodb:<region>:<account-id>:table/<table-name>"
    }
  ]
}
```

# AWS SDK for PHP 버전 3을 사용한 Amazon S3 다중 리전 클라이언트

AWS SDK for PHP 버전 3은 모든 서비스에 사용할 수 있는 일반 다중 리전 클라이언트를 제공합니다. 이 기능을 통해 사용자는 `@region` 입력 파라미터를 명령에 제공하여 명령을 전송할 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. 이 기능은 버킷이 다중 리전에 상주하는 Amazon S3 스트림 래퍼(AWS SDK for PHP 버전 3) (p. 95) 사용자에게 특히 유용합니다.

## 기본 사용법

Amazon S3 클라이언트의 기본 사용 패턴은 표준 S3 클라이언트를 사용하든 다중 리전 클라이언트를 사용하든 상관없이 동일합니다. 명령 수준에서 유일한 사용법 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 상주하는 AWS 리전의 내부 캐시를 유지 관리합니다. 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 Stream Wrapper

Amazon S3 스트림 래퍼를 통해 `file_get_contents`, `fopen`, `copy`, `rename`, `unlink`, `mkdir`, `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 접두사가 오는 문자열(예: `s3://<bucket>[/<key-or-prefix>]`)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 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($steam, 0);
    // Read the same bytes that were previously read
    fread($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. 스트림 래퍼는 다음과 같은 모드를 지원합니다.

| | |
|---|---|
| r | 읽기 전용 스트림이며 파일이 이미 존재해야 합니다. |
| w | 쓰기 전용 스트림이며, 파일이 이미 있는 경우 파일을 덮어씁니다. |
| a | 쓰기 전용 스트림이며, 파일이 이미 있는 경우 파일이 임시 스트림에 다운로드되고, 스트림에 쓰는 내용이 이전에 업로드한 데이터에 추가됩니다. |
| x | 쓰기 전용 스트림이며, 파일이 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.<br>`// Delete an object from a bucket`<br>`unlink('s3://bucket/key');`<br>`DeleteObject` 작업에 사용 가능한 옵션을 전달하여 객체를 삭제하는 방법을 수정할 수 있습니다(예: 특정 객체 버전 지정).<br>`// Delete a specific version of an object from a bucket` |

| | |
|---|---|
| | `unlink('s3://bucket/key', stream_context_create([ 's3' => ['VersionId' => '123'] ]));` |
| filesize() | 객체의 크기를 가져옵니다.<br>`// Get the Content-Length of an object`<br>`$size = filesize('s3://bucket/key', );` |
| is_file() | URL이 파일인지 확인합니다.<br>`if (is_file('s3://bucket/key')) {`<br>`  echo 'It is a file!';`<br>`}` |
| file_exists() | 객체가 있는지 확인합니다.<br>`if (file_exists('s3://bucket/key')) {`<br>`  echo 'It exists!';`<br>`}` |
| 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**
>
> `copy`는 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 작업

PHP를 통해 파일 시스템의 디렉터리를 수정하고 순회할 때와 비슷한 방식으로 Amazon S3 버킷을 수정하고 찾아볼 수 있습니다.

다음은 버킷을 생성하는 예입니다.

```
mkdir('s3://bucket');
```

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://bucket', stream_context_create([
    's3' => ['LocationConstraint' => 'eu-west-1']
]);
```

`rmdir()` 함수를 사용하여 버킷을 삭제할 수 있습니다.

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

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

## 버킷의 내용 나열

Amazon S3 스트림 래퍼에서 opendir(), readdir(), rewinddir() 및 closedir() PHP 함수를 사용하여 버킷 콘텐츠를 순회할 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.

# AWS SDK for PHP 버전 3의 Amazon S3 Transfer Manager

AWS SDK for PHP의 Amazon S3 전송 관리자는 전체 디렉터리를 Amazon S3 버킷에 업로드하고 전체 버킷을 로컬 디렉터리에 다운로드하는 데 사용됩니다.

## Amazon S3에 로컬 디렉터리 업로드

`Aws\S3\Transfer` 객체는 전송을 수행하는 데 사용됩니다. 다음 예에서는 파일의 로컬 디렉터리를 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 source from
$source = '/path/to/source/files';

// Where the files will be transferred to
$dest = 's3://bucket';

// Create a transfer object
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Perform the transfer synchronously
$manager->transfer();
```

이 예에서는 Amazon S3 클라이언트와 `Transfer` 객체를 생성하고 전송을 동기적으로 수행했습니다. 이전 예제를 사용하여 전송을 수행하는 데 필요한 최소한의 코드를 보여줍니다. 전송 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';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->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';

$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Note

버킷에 객체를 다운로드하면 SDK에서 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';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

# 구성

`Transfer` 객체 생성자는 다음 인수를 받습니다.

**`$client`**

전송을 수행하는 데 사용할 `Aws\ClientInterface` 객체입니다.

**`$source`**(string|``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. 반복자를 제공할 경우 `$options` 결합형 배열에 `base_dir` 옵션을 제공해야 합니다.

**`$dest`**

파일이 전송될 대상입니다. `$source` 인수가 디스크에 있는 로컬 경로인 경우 `$dest`는 Amazon S3 버킷 URI(예: `s3://bucket`)이고, `$source` 인수가 Amazon S3 버킷 URI인 경우 `$dest` 인수는 디스크에 있는 로컬 경로여야 합니다.

**`$options`**

전송 옵션 (p. 101)의 결합형 배열입니다.

# 전송 옵션

**`base_dir`** (문자열)

`$source`가 반복자인 경우 소스의 기본 디렉터리입니다. `$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';
$manager = new \Aws\S3\Transfer($client, $source, $dest);

// Initiate the transfer and get a promise
$promise = $manager->promise();

// Do something when the transfer is complete using the then() method
$promise->then(function () {
    echo 'Done!';
});
```

전송되지 않은 파일이 있을 경우 promise가 거부됩니다. promise의 `otherwise` 메서드를 사용하여 실패한 전송을 비동기적으로 처리할 수 있습니다. `otherwise` 함수는 오류 발생 시 호출할 콜백을 받습니다. 콜백은 거부에 대한 `$reason`(일반적으로 `Aws\Exception\AwsException`의 인스턴스)을 받습니다. 하지만 모든 유형의 값을 콜백에 제공할 수 있습니다.

```
$promise->otherwise(function ($reason) {
    echo 'Transfer failed: ';
    var_dump($reason);
});
```

`Transfer` 객체는 promise를 반환하므로 이러한 전송이 다른 비동기적 promise와 동시에 발생할 수 있습니다.

# 전송 관리자의 명령 사용자 지정

생성기에 전달된 콜백을 통해 전송 관리자에서 실행되는 작업에 대해 사용자 지정 옵션을 설정할 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

AWS SDK for PHP는 `S3EncryptionClient`를 제공합니다. 클라이언트 측 암호화를 사용하여 사용자 환경에서 직접 데이터를 암호화 및 암호화 해제합니다. 즉, 이 데이터를 에 전송하기 이전에 암호화하므로 암호화 처리를 위해 외부 서비스를 이용할 필요가 없습니다.

AWS SDK for PHP는 봉투 암호화를 구현하고 OpenSSL을 사용하여 암호화 및 암호화 해제를 수행합니다. 구현은 지원하는 기능이 일치하는 다른 SDK와 상호 연동이 가능합니다. SDK의 promise 기반 비동기 워크플로 (p. 58)와도 호환됩니다.

## 설정

클라이언트 측 암호화를 시작하려면 다음이 필요합니다.

- AWS KMS 암호화 키
- S3 버킷

예제 코드를 실행하기 전에 AWS 자격 증명을 구성합니다. AWD SDK for PHP 버전 3의 자격 증명 (p. 39)을 참조하십시오.

## 암호화

`PutObject` 작업을 통해 암호화된 객체를 업로드할 경우 비슷한 인터페이스가 사용되며 두 개의 새로운 파라미터가 필요합니다.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProvider;

 // Let's construct our S3EncryptionClient using an S3Client
 $encryptionClient = new S3EncryptionClient(
     new S3Client([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ])
 );

 $kmsKeyArn = 'arn-to-the-kms-key';
 // This materials provider handles generating a cipher key and
 // initialization vector, as well as encrypting your cipher key via AWS KMS
 $materialsProvider = new KmsMaterialsProvider(
     new KmsClient([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ]),
     $kmsKeyArn
```

```
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
    // Additional configuration options
];

$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

Note

`'@CipherOptions'`를 올바르게 구성하지 않을 경우 Amazon S3 및 AWS KMS 기반 서비스 오류 이외에 `InvalidArgumentException` 객체가 발생될 수 있습니다.

# 해독

객체를 다운로드하여 암호화 해제하려면 `GetObject` 위에 추가 파라미터가 하나만 필요하며 클라이언트에서 기본 암호 옵션을 자동으로 감지합니다. 추가 구성 옵션은 암호화 해제를 위해 전달됩니다.

```
$result = $encryptionClient->getObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => [
        // Additional configuration options
    ],
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Note

`'@CipherOptions'`를 올바르게 구성하지 않을 경우 Amazon S3 및 AWS KMS 기반 서비스 오류 이외에 `InvalidArgumentException` 객체가 발생될 수 있습니다.

# 암호 구성

**`'Cipher'`** (문자열)

암호화 중에 암호화 클라이언트에서 사용되는 암호 메서드입니다. 여기서는 'gcm' 및 'cbc'만 지원됩니다.

Important

GCM용 OpenSSL 암호화를 사용하여 암호화 및 암호화 해제하는 데 필요한 추가 파라미터를 포함하도록 PHP를 7.1 버전으로 업데이트했습니다. 따라서 PHP 7.1 이상에서만 `Aws\S3\Crypto\S3EncryptionClient`에서 GCM을 사용할 수 있습니다.

**`'KeySize'`** (int)

암호화를 위해 생성할 콘텐츠 암호화 키의 길이입니다. 기본값은 256비트입니다. 유효한 구성 옵션은 256, 192 및 128입니다.

**`'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.

# 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,
    '.instr'
);

$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt'),
]);
```

::class를 호출하여 `HeadersMetadataStrategy` 및 `InstructionFileMetadataStrategy`에 대한 클래스 이름 상수를 제공할 수도 있습니다.

```
$result = $encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => HeadersMetadataStrategy::class,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt'),
]);
```

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\S3EncryptionMultipartUploader`는 업로드 전에 암호화를 위해 소스 스트림을 준비합니다. `Aws\S3\MultipartUploader` 및 `Aws\S3\Crypto\S3EncryptionClient`를 사용할 때와 비슷한 방식으로 생성합니다. `S3EncryptionMultipartUploader`는 `'@MetadataStrategy'`와 동일한 `S3EncryptionClient` 옵션과 모든 사용 가능한 `'@CipherOptions'` 구성을 처리할 수 있습니다.

```
$kmsKeyArn = 'arn-to-the-kms-key';
// This materials provider handles generating a cipher key and
// initialization vector, as well as encrypting your cipher key via AWS KMS
$materialsProvider = new KmsMaterialsProvider(
    new KmsClient([
        'region' => 'us-east-1',
```

```
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyArn
);

$bucket = 'the-bucket-name';
$key = 'the-upload-key';
$cipherOptions = [
    'Cipher' => 'gcm'
    'KeySize' => 256,
    // Additional configuration options
];

$multipartUploader = new S3EncryptionMultipartUploader(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt'),
    [
        '@MaterialsProvider' => $materialsProvider,
        '@CipherOptions' => $cipherOptions,
        'bucket' => 'bucket',
        'key' => 'key',
    ]
);
$multipartUploader->upload();
```

Note

`'@CipherOptions'`를 올바르게 구성하지 않을 경우 Amazon S3 및 AWS KMS 기반 서비스 오류 이외에 `InvalidArgumentException` 객체가 발생될 수 있습니다.

# AWS SDK for PHP 버전 3 코드 예제

AWS SDK for PHP 버전 3에는 SDK를 사용하는 일반적인 Amazon Web Services 시나리오를 보여 주는 코드 예제가 포함되어 있습니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

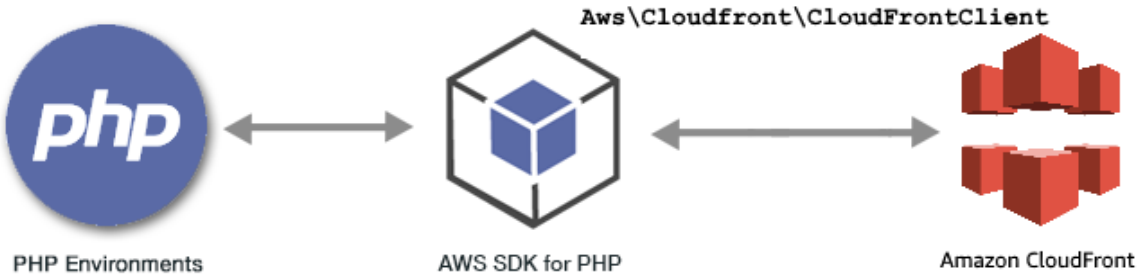
주제



# AWS SDK for PHP 버전 3을 사용하는 Amazon CloudFront 예제

Amazon CloudFront는 웹 서버 또는 AWS 서버(예: Amazon S3)에서 정적/동적 웹 콘텐츠를 더욱 빠르게 공급할 수 있는 AWS 웹 서비스입니다. CloudFront는 엣지 로케이션이라고 하는 전 세계 데이터 센터 네트워크를 통해 콘텐츠를 전송합니다. CloudFront를 사용해 배포하는 콘텐츠를 사용자가 요청하면 콘텐츠가 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 안내서를 참조하십시오.




AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- CloudFront API 및 AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront 배포 관리 (p. 108)
- CloudFront API 및 AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront 무효화 관리 (p. 115)
- AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront URL 서명 (p. 118)

# CloudFront API 및 AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront 배포 관리

Amazon CloudFront는 웹 서버 또는 Amazon 서비스(예: Amazon S3, 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 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서 단원을 참조하십시오.

## 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 해제하십시오.

CloudFront 배포를 생성할 때는 CreateDistribution 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$originName = 'Name to identify the S3 bucket';
$s3BucketURL = '<bucket-name>.s3.amazonaws.com';
$callerReference = 'unique string';
$comment = 'Created by AWS SDK for PHP';
$cacheBehavior = [

    '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,
];

/*
 * $cache = [
 *     'Quantity' => 0,
 * ];
 * $rootObject = '<string>';
 * $alias = [
 *     'Quantity' => 0,
 * ];
 * $customError = [
 *     'Quantity' => 0,
 * ];
 * $httpVersion = 'http1.1';
 * $IPV6 = false;
 * $logging = [
 *     'Bucket' => '',
 *     'Enabled' => false,
 *     'IncludeCookies' => false,
 *     'Prefix' => '',
 * ];
 * $priceClass = 'PriceClass_100';
 * $restrictions = [
 *     'GeoRestriction' => [
 *         'Quantity' => 0,
 *         'RestrictionType' => 'none',
 *     ],
 * ];
 * $viewerCert = [
 *     'CertificateSource' => 'cloudfront',
 *     'CloudFrontDefaultCertificate' => true,
 *     'MinimumProtocolVersion' => 'TLSv1',
 * ];
 * $webACLid = '';
*/


$distribution = [
    'CallerReference' => $callerReference,
    'Comment' => $comment,
    'DefaultCacheBehavior' => $cacheBehavior,
    'Enabled' => $enabled,
    'Origins' => $origin,
    //'CacheBehaviors' => $cache,
    //'DefaultRootObject' => $rootObject,
    //'Aliases' => $alias,
    //'CustomErrorResponses' => $customError,
    //'HttpVersion' => $httpVersion,
    //'IsIPV6Enabled' => $IPV6,
    //'Logging' => $logging,
```

```
    //'PriceClass' => $priceClass,
    //'Restrictions' => $restrictions,
    //'ViewerCertificate' => $viewerCert,
    //'WebACLId' => $webACLid,
];

try {
    $result = $client->createDistribution([
        'DistributionConfig' => $distribution, //REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## CloudFront 배포 가져오기

지정한 CloudFront 배포의 상태와 세부 정보를 가져올 때는 GetDistribution 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$distributionId = 'E1A2B3C4D5E';

try {
    $result = $client->getDistribution([
        'Id' => $distributionId, //REQUIRED
    ]);
    print("<p>The Distribution " . $result['Distribution']['Id'] . " is currently " .
 $result['Distribution']['Status'] . "</p>");
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## CloudFront 배포 나열

현재 계정에서 지정한 AWS 리전에서 기존 CloudFront 배포 목록을 가져올 때는 ListDistributions 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

try {
    $result = $client->listDistributions([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## 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.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);


$id = 'E1A2B3C4D5E';

try {
    $result = $client->getDistribution([
        'Id' => $id,
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

```
$currentConfig = $result["Distribution"]["DistributionConfig"];
$ETag = $result["ETag"];

$distribution = [
    'CallerReference' => $currentConfig["CallerReference"], // REQUIRED
    'Comment' => $currentConfig["Comment"], // REQUIRED
    'DefaultCacheBehavior' => $currentConfig["DefaultCacheBehavior"], // REQUIRED
    'DefaultRootObject' => $currentConfig["DefaultRootObject"],
    'Enabled' => $currentConfig["Enabled"], // REQUIRED
    'Origins' => $currentConfig["Origins"], // REQUIRED
    'Aliases' => $currentConfig["Aliases"],
    'CustomErrorResponses' => $currentConfig["CustomErrorResponses"],
    'HttpVersion' => $currentConfig["HttpVersion"],
    'CacheBehaviors' => $currentConfig["CacheBehaviors"],
    'Logging' => $currentConfig["Logging"],
    'PriceClass' => $currentConfig["PriceClass"],
    'Restrictions' => $currentConfig["Restrictions"],
    'ViewerCertificate' => $currentConfig["ViewerCertificate"],
    'WebACLId' => $currentConfig["WebACLId"],
];


try {
    $result = $client->updateDistribution([
        'DistributionConfig' => $distribution,
        'Id' => $id,
        'IfMatch' => $ETag
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## CloudFront 배포 비활성화

배포를 비활성화하거나 제거하려면 상태를 deployed에서 disabled로 변경합니다.

지정한 CloudFront 배포를 비활성화할 때는 DisableDistribution 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);


$id = 'E1A2B3C4D5E';

try {
```

```
    $result = $client->getDistribution([
        'Id' => $id,
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

$enabled = false;
$currentConfig = $result["Distribution"]["DistributionConfig"];
$ETag = $result["ETag"];

$distribution = [
    'CacheBehaviors' => $currentConfig["CacheBehaviors"], //REQUIRED
    'CallerReference' => $currentConfig["CallerReference"], // REQUIRED
    'Comment' => $currentConfig["Comment"], // REQUIRED
    'DefaultCacheBehavior' => $currentConfig["DefaultCacheBehavior"], // REQUIRED
    'DefaultRootObject' => $currentConfig["DefaultRootObject"],
    'Enabled' => $enabled, // REQUIRED
    'Origins' => $currentConfig["Origins"], // REQUIRED
    'Aliases' => $currentConfig["Aliases"],
    'CustomErrorResponses' => $currentConfig["CustomErrorResponses"],
    'HttpVersion' => $currentConfig["HttpVersion"],
    'IsIPV6Enabled' => $currentConfig["IsIPV6Enabled"],
    'Logging' => $currentConfig["Logging"],
    'PriceClass' => $currentConfig["PriceClass"],
    'Restrictions' => $currentConfig["Restrictions"],
    'ViewerCertificate' => $currentConfig["ViewerCertificate"],
    'WebACLId' => $currentConfig["WebACLId"],
];

//var_dump($distribution);

try {
    $result = $client->updateDistribution([
        'DistributionConfig' => $distribution,
        'Id' => $id,
        'IfMatch' => $ETag
    ]);
    print("<p>For The Distribution " . $result['Distribution']['Id'] . " enabled is set to
 " . $result['Distribution']['DistributionConfig']['Enabled'] . "</p>");
    var_dump($result['Distribution']['DistributionConfig']['Enabled']);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## CloudFront 배포 삭제

배포가 disabled 상태일 때는 해당 배포를 삭제할 수 있습니다.

지정한 CloudFront 배포를 삭제할 때는 DeleteDistribution 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$id = 'E1A2B3C4D5E';

try {
    $result = $client->getDistribution([
        'Id' => $id,
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

$ETag = $result["ETag"];

try {
    $result = $client->deleteDistribution([
        'Id' => $id,
        'IfMatch' => $ETag
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

# CloudFront API 및 AWS SDK for PHP 버전 3을 사용하여 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 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서 단원을 참조하십시오.

## 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.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
// Create a CloudFront Client
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$id = 'E1A2B3C4D5E';
$callerReference = 'STRING';

try {
    $result = $client->createInvalidation([
        'DistributionId' => $id,
        'InvalidationBatch' => [
            'CallerReference' => $callerReference,
            'Paths' => [
                'Items' => ['/*'],
                'Quantity' => 1,
            ],
        ]
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## 배포 무효화 가져오기

CloudFront 배포 무효화에 대한 상태와 세부 정보를 가져올 때는 GetInvalidation 작업을 사용합니다.

가져오기

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

```
use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$distributionId = 'E1A2B3C4D5E';
$invalidationID = 'I1A2B3C4D5E6F7G';

try {
    $result = $client->getInvalidation([
        'DistributionId' => $distributionId,
        'Id' => $invalidationID,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## 배포 무효화 나열

현재 CloudFront 배포 무효화를 모두 나열할 때는 ListInvalidations 작업을 사용합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-2'
]);

$distributionId = 'E1A2B3C4D5E';

try {
    $result = $client->listInvalidations([
        'DistributionId' => $distributionId,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

# AWS SDK for PHP 버전 3을 사용하여 Amazon CloudFront URL 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 Developer Guide의 Amazon CloudFront를 통한 프라이빗 콘텐츠 서비스를 참조하십시오.

- getSignedURL을 사용하여 서명된 Amazon CloudFront URL을 생성합니다.
- getSignedCookie를 사용하여 서명된 Amazon CloudFront 쿠키를 생성합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon CloudFront 사용 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서 단원을 참조하십시오.

## 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.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
// Create a CloudFront Client
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2014-11-06',
    'region' => 'us-east-2'
]);

// Set up parameter values for the resource
$resourceKey = 'rtmp://example-distribution.cloudfront.net/videos/example.mp4';
$expires = time() + 300;

// Create a signed URL for the resource using the canned policy
$signedUrlCannedPolicy = $cloudFront->getSignedUrl([
```

```
    'url' => $resourceKey,
    'expires' => $expires,
    'private_key' => '/path/to/your/cloudfront-private-key.pem',
    'key_pair_id' => '<CloudFront key pair id>'
]);
```

## CloudFront URL 생성 시 사용자 지정 정책 사용

사용자 지정 정책을 사용하려면 `policy` 대신 `expires` 키를 제공합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
// Create a CloudFront Client
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2014-11-06',
    'region' => 'us-east-2'
]);
// Set up parameter values for the resource
$customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
POLICY;

$resourceKey = 'rtmp://example-distribution.cloudfront.net/videos/example.mp4';

// Create a signed URL for the resource using the canned policy
$signedUrlCannedPolicy = $cloudFront->getSignedUrl([
    'url' => $resourceKey,
    'policy' => $customPolicy,
    'private_key' => '/path/to/your/cloudfront-private-key.pem',
    'key_pair_id' => '<CloudFront key pair id>'
]);
foreach ($signedUrlCannedPolicy  as $name => $value) {
    setcookie($name, $value, 0, "", "example-distribution.cloudfront.net", true, true);
}
```

## 서명된 CloudFront 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 있습니다. 서명된 URL 대신 서명된 쿠키를 사용해야 하는 이유(또는 반대의 경우)에 대한 자세한 내용은 Amazon CloudFront Developer Guide의 서명된 URL과 서명된 쿠키 간의 선택을 참조하십시오.

서명된 쿠키를 생성하는 방법은 서명된 URL을 생성하는 방법과 비슷합니다. 유일한 차이점은 호출되는 메서드입니다(`getSignedUrl` 대신 `getSignedCookie`).

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
// Create a CloudFront Client
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2014-11-06',
    'region' => 'us-east-2'
]);

// Set up parameter values for the resource
$resourceKey = 'https://example-distribution.cloudfront.net/videos/example.mp4';
$expires = time() + 300;

// Create a signed cookie for the resource using the canned policy
$signedCookieCannedPolicy = $cloudFront->getSignedCookie([
    'url' => $resourceKey,
    'expires' => $expires,
    'private_key' => '/path/to/your/cloudfront-private-key.pem',
    'key_pair_id' => '<CloudFront key pair id>'
]);
```

## CloudFront 쿠키 생성 시 사용자 지정 정책 사용

`getSignedUrl`과 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.

`getSignedCookie`는 프라이빗 배포에 대한 액세스 권한을 부여하기 위해 모두 쿠키로 설정해야 하는 키-값 페어의 배열을 반환합니다.

가져오기

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

use Aws\CloudFront\CloudFrontClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
// Create a CloudFront Client
$client = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2014-11-06',
    'region' => 'us-east-2'
]);

// Set up parameter values for the resource
$resourceKey = 'https://example-distribution.cloudfront.net/videos/example.mp4';
$customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
POLICY;

// Create a signed cookie for the resource using a custom policy
$signedCookieCustomPolicy = $cloudFront->getSignedCookie([
    'policy' => $customPolicy,
    'private_key' => '/path/to/your/cloudfront-private-key.pem',
    'key_pair_id' => '<CloudFront key pair id>'
]);

foreach ($signedCookieCustomPolicy as $name => $value) {
    setcookie($name, $value, 0, "", "example-distribution.cloudfront.net", true, true);
}
```

## 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 내용은 Amazon CloudFront Developer Guide의 서명된 쿠키 사용을 참조하십시오.

# AWS SDK for PHP 버전 3을 사용하여 사용자 지정 Amazon CloudSearch 도메인 요청 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 자격 증명 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## CSlong 도메인 요청에 서명

가져오기

```
use Aws\Credentials\CredentialProvider;
use Aws\Signature\SignatureV4;
use GuzzleHttp\Client;
use GuzzleHttp\Psr7\Request;
```

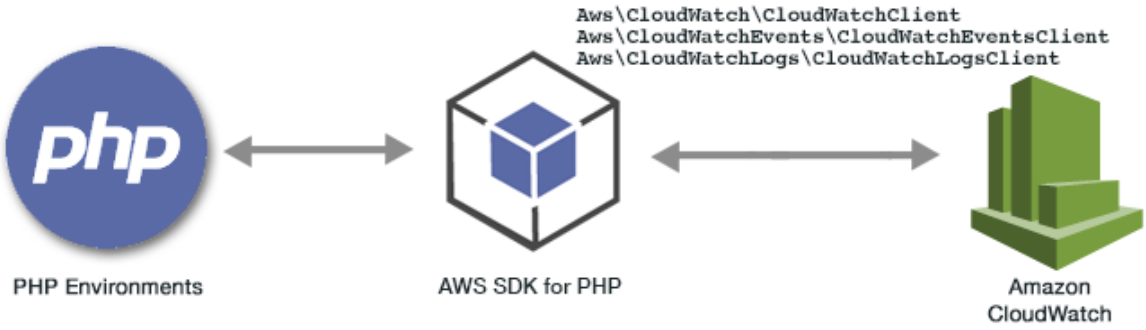
샘플 코드

```
// Prepare a CloudSearch domain request
$request = new Request(
    'GET',
```

```
    'https://<your-domain>.<region-of-domain>.cloudsearch.amazonaws.com/2013-01-01/search?
q=star+wars&return=title'
);

// Get your credentials from the environment
$credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

// Construct a request signer
$signer = new SignatureV4('cloudsearch', '<region-of-domain>');

// Sign the request
$request = $signer->signRequest($request, $credentials);

// Send the request
$response = (new Client)->send($request);
$results = json_decode($response->getBody());
if ($results->hits->found > 0) {
    echo $results->hits->hit[0]->fields->title . "\n";
}
```

# AWS SDK for PHP 버전 3을 사용하는 Amazon CloudWatch 예제

Amazon CloudWatch(CloudWatch)는 Amazon Web Services(AWS) 리소스와 AWS에서 실행하는 애플리케이션을 실시간으로 모니터링하는 웹 서비스입니다. 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3으로 Amazon CloudWatch 경보 작업 (p. 124)
- AWS SDK for PHP 버전 3으로 Amazon CloudWatch에서 지표 가져오기 (p. 126)
- AWS SDK for PHP 버전 3으로 Amazon CloudWatch에서 사용자 지정 지표 게시 (p. 128)
- AWS SDK for PHP 버전 3으로 Amazon CloudWatch Events에 이벤트 전송 (p. 130)
- AWS SDK for PHP 버전 3으로 Amazon CloudWatch 경보에서 경보 작업 사용 (p. 132)

# AWS SDK for PHP 버전 3으로 Amazon CloudWatch 경보 작업

Amazon CloudWatch 경보는 지정한 기간 동안 단일 지표를 감시합니다. 경보는 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 경보 설명

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->describeAlarms([
    ]);
    foreach ($result['MetricAlarms'] as $alarm) {
        echo $alarm['AlarmName'] . "\n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 경보 만들기

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->putMetricAlarm(array(
        // AlarmName is required
        'AlarmName' => 'string',
        // MetricName is required
        'MetricName' => 'string',
        // Namespace is required
        'Namespace' => 'string',
        // Statistic is required
        //string: SampleCount | Average | Sum | Minimum | Maximum
        'Statistic' => 'string',
        // Period is required
        'Period' => integer,
        'Unit' => 'Count/Second',
        // EvaluationPeriods is required
        'EvaluationPeriods' => integer,
        // Threshold is required
        'Threshold' => integer,
        // ComparisonOperator is required
        // string: GreaterThanOrEqualToThreshold | GreaterThanThreshold | LessThanThreshold
 | LessThanOrEqualToThreshold
        'ComparisonOperator' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# 경보 삭제

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);
```

```
try {
    $result = $client->deleteAlarms([
        'AlarmNames' => [$alarmName] // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3으로 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 지표 나열

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->listMetrics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 지표에 대한 경보 검색

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->describeAlarmsForMetric(array(
        // MetricName is required
        'MetricName' => 'ApproximateNumberOfMessagesVisible',
        // Namespace is required
        'Namespace' => 'AWS/SQS',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 지표 통계 가져오기

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->getMetricStatistics(array(
        'Namespace' => 'string',
        'MetricName' => 'CloudWatchTests',
        //StartTime : mixed type: string (date format)|int (unix timestamp)|\DateTime
        'StartTime' => strtotime('-1 days'),
```

```
        //EndTime : mixed type: string (date format)|int (unix timestamp)|\DateTime
        'EndTime' => strtotime('now'),
        //The granularity, in seconds, of the returned datapoints. Period must be at least
 60 seconds and must be a multiple of 60. The default value is 60
        'Period' => 3000,
        'Statistics' => array('Maximum', 'Minimum'),
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3으로 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 지표 데이터 게시

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->putMetricData(array(
        'Namespace' => 'string',
        'MetricData' => array(
            array(
```

```
                'MetricName' => 'string',
                //Timestamp : mixed type: string (date format)|int (unix timestamp)|
\DateTime
                'Timestamp' => time(),
                'Value' => integer,
                'Unit' => 'Kilobytes'
            )
        )
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# 경보 만들기

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->putMetricAlarm(array(
        // AlarmName is required
        'AlarmName' => 'string',
        // MetricName is required
        'MetricName' => 'string',
        // Namespace is required
        'Namespace' => 'string',
        // Statistic is required
        //string: SampleCount | Average | Sum | Minimum | Maximum
        'Statistic' => 'string',
        // Period is required
        'Period' => integer,
        'Unit' => 'Count/Second',
        // EvaluationPeriods is required
        'EvaluationPeriods' => integer,
        // Threshold is required
        'Threshold' => integer,
        // ComparisonOperator is required
        // string: GreaterThanOrEqualToThreshold | GreaterThanThreshold | LessThanThreshold
 | LessThanOrEqualToThreshold
        'ComparisonOperator' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3으로 Amazon CloudWatch Events에 이벤트 전송

CloudWatch 이벤트는 Amazon Web Services(AWS)의 변경 내용을 설명하는 실시간에 가까운 시스템 이벤트 스트림을 다양한 대상으로 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 전송합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 규칙 생성

가져오기

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

use Aws\CloudWatchEvents\CloudWatchEventsClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\cloudwatchevents\cloudwatcheventsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2015-10-07'
]);

try {
    $result = $client->putRule(array(
        'Name' => 'DEMO_EVENT', // REQUIRED
        'RoleArn' => 'IAM_ROLE_ARN',
        'ScheduleExpression' => 'rate(5 minutes)',
        'State' => 'ENABLED',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 규칙에 대상 추가

가져오기

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

use Aws\CloudWatchEvents\CloudWatchEventsClient;
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

가져오기

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

use Aws\CloudWatchEvents\CloudWatchEventsClient;
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 3으로 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 경보 작업 활성화

가져오기

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

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->enableAlarmActions([
        'AlarmNames' => array($alarmName) //REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

## 경보 작업 비활성화

가져오기

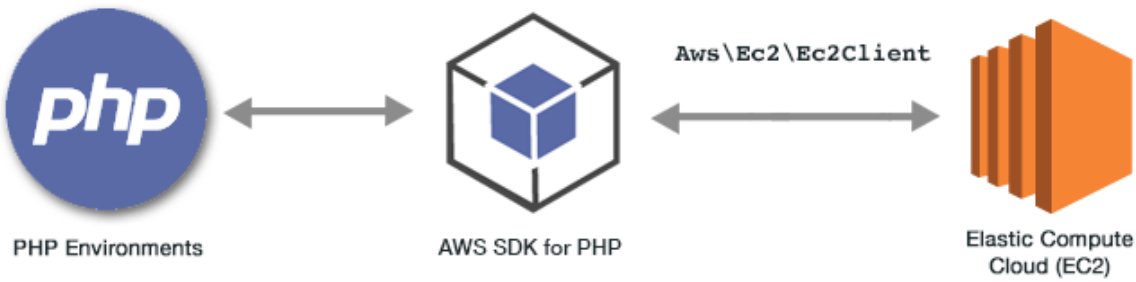
```
require 'vendor/autoload.php';

use Aws\CloudWatch\CloudWatchClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new Aws\CloudWatch\CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2010-08-01'
]);

try {
    $result = $client->disableAlarmActions([
        'AlarmNames' => array($alarmName) //REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3을 사용하는 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 클라우드 컴퓨팅을 수행할 수 있도록 설계되었습니다.




AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

# 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

주제



# AWS SDK for PHP 버전 3을 사용하여 Amazon EC2 인스턴스 관리

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeInstances를 사용하여 Amazon EC2 인스턴스를 설명합니다.
- 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 인스턴스 설명

가져오기

```
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();
```

```
var_dump($result);
```

## 모니터링 활성화 및 비활성화

가져오기

```
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'
]);

$instanceIds = array('InstanceID1', 'InstanceID2');

$monitorInstance = 'ON';

if ($monitorInstance == 'ON') {
    $result = $ec2Client->monitorInstances(array(
        'InstanceIds' => $instanceIds
    ));
} else {
    $result = $ec2Client->unmonitorInstances(array(
        'InstanceIds' => $instanceIds
    ));
}

var_dump($result);
```

## 인스턴스 시작 및 중지

가져오기

```
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'
]);

$action = 'START';

$instanceIds = array('InstanceID1', 'InstanceID2');
```

```
if ($action == 'START') {
    $result = $ec2Client->startInstances(array(
        'InstanceIds' => $instanceIds,
    ));
} else {
    $result = $ec2Client->stopInstances(array(
        'InstanceIds' => $instanceIds,
    ));
}

var_dump($result);
```

## 인스턴스 재부팅

가져오기

```
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'
]);

$instanceIds = array('InstanceID1', 'InstanceID2');

$result = $ec2Client->rebootInstances(array(
    'InstanceIds' => $instanceIds
));

var_dump($result);
```

# AWS SDK for PHP 버전 3으로 Amazon EC2에서 탄력적 IP 주소 사용

탄력적 IP 주소는 동적 클라우드 컴퓨팅을 위해 고안된 고정 IP 주소입니다. 탄력적 IP 주소는 AWS 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 릴리스합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 인스턴스 설명

가져오기

```
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();

var_dump($result);
```

## 주소 할당 및 연결

가져오기

```
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'
]);

$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

가져오기

```
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'
]);

$associationID = 'AssociationID';

$allocationID = 'AllocationID';

$result = $ec2Client->disassociateAddress(array(
    'AssociationId' => $associationID,
));

$result = $ec2Client->releaseAddress(array(
    'AllocationId' => $allocationID,
));

var_dump($result);
```

# AWS SDK for PHP 버전 3으로 Amazon EC2에서 리전 및 가용 영역 사용

Amazon EC2는 전 세계 여러 위치에서 호스팅되고 있습니다. 이 위치들은 AWS 리전과 가용 영역으로 구성됩니다. 각 리전은 가용 영역이라는 격리된 위치가 여러 개 있는 지리적 개별 영역입니다. Amazon EC2는 여러 위치에 인스턴스와 데이터를 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 가용 영역 설명

가져오기

```
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->describeAvailabilityZones();

var_dump($result);
```

## 리전 설명

가져오기

```
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->describeRegions();

var_dump($result);
```

# AWS SDK for PHP 버전 3으로 Amazon EC2 키 페어 작업

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 키 페어 생성

가져오기

```
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'
]);

$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

가져오기

```
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'
```

```
]);

$keyPairName = 'my-keypair';

$result = $ec2Client->deleteKeyPair(array(
    'KeyName' => $keyPairName
));

var_dump($result);
```

## 키 페어 설명

가져오기

```
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->describeKeyPairs();

var_dump($result);
```

# AWS SDK for PHP 버전 3으로 Amazon EC2에서 보안 그룹 작업

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 보안 그룹 설명

가져오기

```
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->describeSecurityGroups();

var_dump($result);
```

## 수신 규칙 추가

가져오기

```
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->authorizeSecurityGroupIngress(array(
    'GroupName' => 'string',
    'SourceSecurityGroupName' => 'string'
));

var_dump($result);
```

## 보안 그룹 생성

가져오기

```
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'
]);

// 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

가져오기

```
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'
]);

$securityGroupId = 'my-security-group-id';

$result = $ec2Client->deleteSecurityGroup(array(
    'GroupId' => $securityGroupId
));

var_dump($result);
```

# AWS SDK for PHP 버전 3을 사용하여 Amazon Elasticsearch Service 검색 요청 서명

Amazon Elasticsearch Service(Amazon ES)는 널리 사용되는 오픈 소스 검색 및 분석 엔진인 Amazon Elasticsearch Service를 손쉽게 배포, 운영 및 확장할 수 있게 해주는 관리형 서비스입니다. Amazon ES는 Amazon Elasticsearch Service API에 대한 직접 액세스를 제공합니다. 따라서 개발자는 친숙한 도구뿐 아니라 IAM 사용자 및 역할을 액세스 제어에 사용하는 것과 같은 확실한 보안 옵션을 사용할 수 있습니다. 많은 Amazon Elasticsearch Service 클라이언트는 요청 서명을 지원하지만, 지원하지 않는 클라이언트를 사용하는 경우 AWS SDK for PHP의 내장 자격 증명 공급자 및 서명자를 사용하여 임의의 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## Amazon ES 요청 서명

Amazon ES는 서명 버전 4를 사용합니다. 따라서 서비스의 서명 이름(이 경우 `es`)과 Amazon ES 도메인의 AWS 리전에 대해 요청에 서명해야 합니다. Amazon ES에서 지원하는 전체 리전 목록은 Amazon Web Services General Reference의 AWS 리전 및 엔드포인트 페이지에서 확인할 수 있습니다. 하지만 이번 예제에서는 `us-west-2` 리전의 Amazon ES 도메인에 대해 요청에 서명합니다.

자격 증명을 제공해야 하며, SDK의 기본 공급자 체인 또는 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 모든 형식의 자격 증명을 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 사용하는 AWS IAM 예제

AWS Identity and Access Management(IAM)는 Amazon Web Services(AWS) 고객이 AWS에서 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. 을 통해 사용자, 액세스 키와 같은 보안 자격 증명, 사용자가 어떤 AWS 리소스에 액세스할 수 있는지 제어하는 권한을 한 곳에서 관리할 수 있습니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

# 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용하여 IAM 액세스 키 관리 (p. 145)
- AWS SDK for PHP 버전 3을 사용하여 IAM 사용자 관리 (p. 148)
- AWS SDK for PHP 버전 3을 통해 IAM 계정 별칭 사용 (p. 151)
- AWS SDK for PHP 버전 3을 사용하여 IAM 정책 작업 (p. 153)
- AWS SDK for PHP 버전 3을 사용하여 IAM 서버 인증서 작업 (p. 160)

# AWS SDK for PHP 버전 3을 사용하여 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 페어를 삭제합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 액세스 키 생성

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

```
use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

```
```

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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 3을 사용하여 IAM 사용자 관리

IAM 사용자는 AWS에서 생성하는 개체로서 AWS와 상호 작용하기 위해 그 개체를 사용하는 사람 또는 서비스를 대표합니다. AWS에서 사용자는 이름과 자격 증명으로 구성됩니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateUser를 사용하여 새 IAM 사용자를 생성합니다.
- ListUsers를 사용하여 IAM 사용자의 목록을 표시합니다.
- UpdateUser를 사용하여 IAM 사용자를 업데이트합니다.
- GetUser를 사용하여 IAM 사용자에 대한 정보를 검색합니다.
- DeleteUser를 사용하여 IAM 사용자를 삭제합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## IAM 사용자 생성

가져오기

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

use Aws\Iam\IamClient;
```

```
use Aws\Exception\AwsException;
```

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 목록 표시

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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 업데이트

가져오기

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

use Aws\Iam\IamClient;
```

```
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->updateUser(array(
        // UserName is required
        'UserName' => 'string1',
        'NewUserName' => 'string'
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## IAM 사용자 정보 조회

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->getUser(array(
        'UserName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## IAM 사용자 삭제

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->deleteUser(array(
        // UserName is required
        'UserName' => 'string'
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3을 통해 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 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 별칭 만들기

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->deleteAccountAlias(array(
        // AccountAlias is required
        'AccountAlias' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3을 사용하여 IAM 정책 작업

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 정책 만들기

가져오기

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

use Aws\Iam\IamClient;
```

```
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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());
}
```

## 사용자에 정책 연결

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->detachUserPolicy(array(
        // UserName is required
        'UserName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 그룹 정책 분리

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->detachGroupPolicy(array(
        // GroupName is required
        'GroupName' => 'string',
        // PolicyArn is required
        'PolicyArn' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 정책 삭제

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->deleteRolePolicy(array(
        // RoleName is required
        'RoleName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string'
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

## 사용자 정책 삭제

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->deleteUserPolicy(array(
        // UserName is required
        'UserName' => 'string',
        // PolicyName is required
        'PolicyName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 그룹 정책 삭제

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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 사용하여 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 인증서 목록을 표시합니다.
- GetServerCertificate를 사용하여 인증서에 대한 정보를 검색합니다.
- UpdateServerCertificate를 사용하여 인증서를 업데이트합니다.
- DeleteServerCertificate를 사용하여 인증서를 삭제합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 서버 인증서 나열

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->getServerCertificate(array(
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 서버 인증서 업데이트

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->updateServerCertificate(array(
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
        'NewServerCertificateName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

## 서버 인증서 삭제

가져오기

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

use Aws\Iam\IamClient;
use Aws\Exception\AwsException;
```

샘플 코드

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

try {
    $result = $client->deleteServerCertificate(array(
        // ServerCertificateName is required
        'ServerCertificateName' => 'string',
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3을 사용하는 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 참조하십시오. 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 사용하여 키 작업

AWS Key Management Service(AWS KMS)의 기본 리소스는 고객 마스터 키(CMK)입니다. CMK를 사용하여 데이터를 암호화할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateKey를 사용하여 고객 CMK를 생성합니다.
- GenerateDataKey를 사용하여 데이터 키를 생성합니다.
- DescribeKey를 사용하여 CMK를 봅니다.
- ListKeys를 사용하여 CMK의 키 ID와 키 ARN을 확인합니다 .
- EnableKey를 사용하여 CMK 활성화.
- using DisableKey를 사용하여 CMK를 비활성화합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하십시오.

## CMK를 생성합니다

CMK를 생성하려면 CreateKey 운영에 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK(고객 마스터 키)를 지정합니다.

가져오기

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

use Aws\Kms\KmsClient;
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";
}
```

## CMK 보기

CMK의 Amazon 리소스 이름(ARN) 및 키 상태를 비롯해 고객 마스터 키(CMK)에 대한 자세한 정보를 가져오려면 DescribeKey 작업을 사용합니다.

`DescribeKey`는 별칭을 확인하지 않습니다. 별칭을 가져오려면 ListAliases 작업을 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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";
}
```

## CMK의 키 ID와 키 ARN 확인

CMK의 ID와 ARN을 확인하려면 ListAliases 작업을 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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";
}
```

## CMK 활성화

비활성화된 CMK를 활성화하려면 EnableKey 작업을 이용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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";
}
```

## CMK 비활성화

CMK를 비활성화하려면 DisableKey 작업을 이용합니다. CMK를 비활성화하면 삭제할 수 없습니다.

가져오기

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

use Aws\Kms\KmsClient;
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 SDK for PHP 버전 3을 사용하여 AWS KMS 데이터 키 암호화

데이터 키는 많은 양의 데이터 및 기타 데이터 암호화 키를 포함하여 데이터를 암호화하는 데 사용할 수 있는 암호화 키입니다.

데이터 키의 생성, 암호화 및 암호화 해제를 위해 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용할 수 있습니다. 그러나 AWS KMS는 데이터 키를 저장, 관리 또는 추적하거나 데이터 키로 암호화 작업을 수행하지 않습니다. 따라서 AWS KMS 밖에서 데이터 키를 사용하고 관리합니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- 암호화를 사용하여 데이터 키를 암호화합니다.
- 암호화 해제를 사용하여 데이터 키를 암호화 해제합니다.
- ReEncrypt를 사용하여 새로운 CMK로 데이터 키를 다시 암호화합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하십시오.

## 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 CMK를 사용하여 데이터 키를 암호화하려는 경우 `Encypt` 메서드를 사용할 수 있습니다.

가져오기

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

use Aws\Kms\KmsClient;
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";
}
```

## 암호화 해제

데이터 키를 해독하려면 Decrypt 작업을 사용합니다.

설정된 `ciphertextBlob`은 GenerateDataKey, GenerateDataKeyWithoutPlaintext 또는 Encrypt 응답에서 나온 `CiphertextBlob` 필드의 값이어야 합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK에서 즉시 데이터 키를 재암호화하려면 ReEncrypt 작업을 사용합니다. 이러한 작업은 모두 AWS KMS 내부의 서버 측에서 수행되므로 AWS KMS 외부에 일반 텍스트를 노출해서는 안 됩니다.

설정된 `ciphertextBlob`은 GenerateDataKey, GenerateDataKeyWithoutPlaintext 또는 Encrypt 응답에서 나온 `CiphertextBlob` 필드의 값이어야 합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 KMS API 및 AWS SDK for PHP 버전 3을 사용하여 AWS KMS 키 정책 작업

AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 생성할 때 CMK를 사용하고 관리할 수 있는 사람을 지정합니다. 이러한 권한은 키 정책이라는 문서에 포함됩니다. 키 정책을 사용하여 고객 관리형

CMK에 대한 권한을 언제든 추가, 삭제하거나 수정할 수 있지만 AWS에서 관리하는 CMK에 대한 키 정책은 편집할 수 없습니다. 자세한 내용은 AWS KMS 인증 및 액세스 제어를 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하십시오.

## 모든 키 정책의 목록 표시

CMK의 키 정책 이름을 확인하려면 `ListKeyPolicies` 작업을 사용합니다. 반환되는 유일한 키 정책 이름은 기본 이름입니다.

가져오기

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

use Aws\Kms\KmsClient;
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

CMK의 키 정책을 확인하려면 `GetKeyPolicy` 작업을 사용합니다.

`GetKeyPolicy`는 정책 이름을 요구합니다. CMK를 생성할 때 키 정책을 생성하지 않은 경우 유효한 정책 이름은 기본 이름뿐입니다. 기본 키 정책에 대해 자세히 알아보십시오.

가져오기

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

use Aws\Kms\KmsClient;
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

CMK에 대한 키 정책을 설정하거나 변경하려면 `PutKeyPolicy` 작업을 사용합니다.

`PutKeyPolicy`는 정책 이름을 요구합니다. CMK를 생성할 때 키 정책을 생성하지 않은 경우 유효한 정책 이름은 기본 이름뿐입니다. 기본 키 정책에 대해 자세히 알아보십시오.

가져오기

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

use Aws\Kms\KmsClient;
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 사용하여 권한 부여 작업

권한 부여는 키 정책의 대안으로서 권한을 제공하는 또 하나의 메커니즘입니다. 권한 부여를 이용해 AWS 보안 주체가 AWS Key Management Service(AWS KMS) 고객 관리형 CMK를 사용하도록 허용하는 장기 액세스를 부여할 수 있습니다. 자세한 내용은 권한 부여 사용을 참조하십시오.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateGrant를 사용하여 고객 마스터 키(CMK)에 대한 권한 부여를 생성합니다.
- ListGrants를 사용하여 CMK에 대한 권한 부여를 봅니다.
- RetireGrant를 사용하여 CMK에 대한 권한 부여를 만료시킵니다.
- RevokeGrant를 사용하여 CMK에 대한 권한 부여를 취소합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하십시오.

## 권한 생성

AWS KMS CMK에 대한 권한 부여를 생성하려면 CreateGrant 작업을 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK에 대한 권한 부여를 자세히 알아보려면 ListGrants 작업을 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK에 대한 권한 부여를 만료시키려면 RetireGrant 작업을 사용합니다. 권한 부여의 사용을 완료한 후에는 만료시킵니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK에 대한 권한 부여를 취소하려면 RevokeGrant 작업을 사용합니다. 권한 부여를 취소하여 종속된 작업을 명시적으로 거부할 수 있습니다.

가져오기

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

use Aws\Kms\KmsClient;
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 사용하여 별칭 작업

별칭은 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)에 대한 선택적 표시 이름입니다.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

AWS Key Management Service(AWS KMS) 사용에 대한 자세한 정보는 AWS KMS 개발자 안내서를 참조하십시오.

## 별칭 만들기

CMK에 대한 별칭을 생성하려면 CreateAlias 작업을 사용합니다. 별칭은 계정 및 AWS 리전 내에서 고유해야 합니다. 이미 별칭이 있는 CMK에 별칭을 생성할 경우, `CreateAlias`가 동일한 CMK에 대해 다른 별칭을 생성합니다. 새 별칭이 기존 별칭을 대체하지는 않습니다.

가져오기

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

use Aws\Kms\KmsClient;
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

별칭을 나열하려면 ListAliases 작업을 사용합니다. AWS 서비스가 정의하지만 CMK와 연결되지 않은 별칭이 응답에 포함됩니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK와 연결하려면 UpdateAlias 작업을 사용합니다.

가져오기

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

use Aws\Kms\KmsClient;
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 CMK에 영향을 미치지 않습니다.

가져오기

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

use Aws\Kms\KmsClient;
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 사용하는 Amazon Kinesis 예제

Amazon Kinesis는 데이터를 실시간으로 수집, 처리, 분석하는 AWS 서비스입니다. Amazon Kinesis Data Streams를 사용하여 데이터 스트림을 구성하거나, Amazon Kinesis Data Firehose를 사용하여 Amazon S3, Amazon ES, Amazon Redshift 또는 Splunk로 데이터를 전송합니다.

Kinesis에 대한 자세한 내용은 Amazon Kinesis 문서를 참조하십시오.


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 (p. 179)
- Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 샤드 관리 (p. 183)
- Kinesis Data Firehose API 및 AWS SDK for PHP 버전 3을 사용하여 전송 스트림 생성 (p. 185)

# Kinesis Data Streams API 및 AWS SDK for PHP 버전 3을 사용하여 데이터 스트림 생성

Amazon Kinesis Data Streams는 실시간 데이터를 전송하도록 허용합니다. Kinesis Data Streams를 사용하여 데이터를 추가할 때마다 구성한 대상으로 데이터를 전송하는 데이터 생산자를 생성합니다.

자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 스트림 생성 및 관리를 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon Kinesis Data Streams 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서를 참조하십시오.

## Kinesis Data Stream을 사용하여 데이터 스트림 생성

다음 코드 예제를 사용하여 Kinesis에서 처리할 정보를 전송할 수 있는 Kinesis 데이터 스트림을 설정합니다. 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 데이터 스트림 생성 및 업데이트를 참조하십시오.

Kinesis 데이터 스트림을 생성하려면, CreateStream 작업을 사용하십시오.

가져오기

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

use Aws\Kinesis\KinesisClient;
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 확인하십시오.

지정한 Kinesis 데이터 스트림에 대한 세부 정보를 확인하려면 DescribeStream 작업을 사용합니다.

가져오기

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

use Aws\Kinesis\KinesisClient;
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 계정의 처음 10개 데이터 스트림의 목록을 표시합니다. 반환된 `HasMoreStreams`를 사용하여 계정에 더 많은 스트림이 연결되었는지를 확인합니다.

Kinesis 데이터 스트림의 목록을 표시하려면 ListStreams 작업을 사용합니다.

가져오기

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

use Aws\Kinesis\KinesisClient;
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 `DescribeStream`을 사용하여 데이터의 `StreamStatus`가 활성인지 여부를 확인합니다.

Kinesis 데이터 스트림에 단일 데이터 레코드를 기록하려면 PutRecord 작업을 사용합니다. Kinesis 데이터 스트림에 최대 500개의 레코드를 기록하려면 PutRecords 작업을 사용합니다.

가져오기

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

```
use Aws\Kinesis\KinesisClient;
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.

가져오기

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

use Aws\Kinesis\KinesisClient;
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 실시간 데이터를 엔드포인트로 보낼 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 관리하십시오.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- ListShards를 사용하여 스트림의 샤드 목록을 표시합니다.
- UpdateShardCount를 사용하여 스트림의 샤드 수를 추가하거나 줄입니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon Kinesis Data Streams 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서를 참조하십시오.

## 데이터 스트림 샤드 목록 표시

특정 스트림의 최대 100개 샤드에 대한 세부 정보를 표시합니다.

Kinesis 데이터 스트림의 샤드 목록을 표시하려면 ListShards 작업을 사용합니다.

가져오기

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

use Aws\Kinesis\KinesisClient;
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 데이터 스트림의 샤드 수를 업데이트하려면 UpdateShardCount 작업을 사용합니다.

가져오기

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

use Aws\Kinesis\KinesisClient;
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";
}
```

# Kinesis Data Firehose API 및 AWS SDK for PHP 버전 3을 사용하여 전송 스트림 생성

Amazon Kinesis Data Firehose를 사용하면 Amazon Kinesis Data Streams, Amazon S3, Amazon Elasticsearch Service (Amazon ES), Amazon Redshift 등의 AWS 서비스로 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon Kinesis Data Firehose 사용에 대한 자세한 내용은 Amazon Kinesis Data Firehose 개발자 안내서를 참조하십시오.

## 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 Kinesis Data Firehose로 마이그레이션할 수 있습니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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 대상에 대한 액세스 권한 부여에서 설명한 대로 Amazon S3 버킷에 대한 액세스 권한을 Kinesis Data Firehose에 부여해야 합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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";
}
```

## Amazon ES를 사용하여 전송 스트림을 생성하려면

데이터를 Amazon ES 클러스터에 넣는 Kinesis Data Firehose 전송 스트림을 설정하려면 CreateDeliveryStream 작업을 사용합니다.

대상 파라미터에서 설명한 대로 대상 파라미터를 제공합니다. Kinesis Data Firehose에 Amazon ES 대상에 대한 액세스 권한 부여에서 설명한 대로 Amazon ES 클러스터에 대한 액세스 권한을 Kinesis Data Firehose에 부여해야 합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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 Kinesis Data Firehose 전송 시스템에 대한 세부 정보를 확인하려면 DescribeDeliveryStream 작업을 사용합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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

Kinesis Data Streams로 데이터를 전송하는 기존의 모든 Kinesis Data Firehose 전송 스트림 목록을 표시하려면, ListDeliveryStreams 작업을 사용합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
```

```
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, Amazon ES, Amazon Redshift로 또는 Splunk로 데이터를 전송하는 기존의 모든 Kinesis Data Firehose 전송 스트림 목록을 표시하려면, ListDeliveryStreams 작업을 사용합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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 Kinesis Data Firehose 전송 스트림에 데이터 전송

Kinesis Data Firehose 전송 스트림을 통해 지정한 대상으로 데이터를 전송하려면 Kinesis Data Firehose 전송 스트림을 생성한 후 PutRecord 작업을 사용합니다.

Kinesis Data Firehose 전송 스트림으로 데이터를 전송하기 전에 `DescribeDeliveryStream`을 사용하여 전송 스트림이 활성 상태인지 확인합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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";
}
```

## Kinesis Data Firehose 전송 스트림 삭제

Kinesis Data Firehose 전송 스트림을 삭제하려면 DeleteDeliveryStreams 작업을 사용합니다. 이 작업은 전송 스트림으로 전송한 데이터도 삭제합니다.

가져오기

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

use Aws\Firehose\FirehoseClient;
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 사용하는 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 참조하십시오.


AWS Elemental MediaConvert용 PHP API는 AWS.MediaConvert 클라이언트 클래스를 통해 노출됩니다. 자세한 내용은 API 참조의 Class: AWS.MediaConvert를 참조하십시오.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3으로 AWS Elemental MediaConvert를 위한 계정별 엔드포인트 가져오기 (p. 192)
- AWS SDK for PHP 버전 3으로 AWS Elemental MediaConvert에서 트랜스코딩 작업 생성 및 관리 (p. 193)

# AWS SDK for PHP 버전 3으로 AWS Elemental MediaConvert를 위한 계정별 엔드포인트 가져오기

이 예제에서는 AWS SDK for PHP 버전 3을 사용하여 AWS Elemental MediaConvert를 호출한 후 계정별 엔드포인트를 검색합니다. 서비스 기본 엔드포인트에서 엔드포인트 URL을 검색할 수 있으며 이 작업을 수행하는 데는 계정별 엔드포인트가 필요하지 않습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- DescribeEndpoints를 사용하여 엔드포인트를 검색합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 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 사용 설명서의 IAM 권한 설정을 참조하십시오.

## 엔드포인트 검색

`AWS.MediaConvert` 클라이언트 클래스의 describeEndpoints 메서드에 대한 빈 요청 파라미터를 전달할 객체를 생성합니다. describeEndpoints 메서드를 호출하려면 AWS Elemental MediaConvert 서비스 객체를 호출하여 파라미터를 전달하기 위한 promise를 생성합니다. promise 콜백에서 응답을 처리합니다.

가져오기

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

use Aws\MediaConvert\MediaConvertClient;
use Aws\Exception\AwsException;
```

샘플 코드

엔드포인트를 가져올 리전을 정의하고 MediaConvert 클라이언트 객체를 생성합니다.

```
$client = new Aws\MediaConvert\MediaConvertClient([
    'profile' => 'default',
    'version' => '2017-08-29',
    'region' => 'us-east-2'
]);

//retrieve endpoint
try {
    $result = $client->describeEndpoints([]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

describeEndpoints 메서드를 호출하여 엔드포인트를 검색하고 엔드포인트의 URL을 저장합니다.

```
$single_endpoint_url = $result['Endpoints'][0]['Url'];

print("Your endpoint is " . $single_endpoint_url);

//Create an AWSMediaConvert client object with the endpoint URL that you retrieved:
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default',
    'endpoint' => $single_endpoint_url
]);
```

# AWS SDK for PHP 버전 3으로 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 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 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 사용 설명서의 IAM 권한 설정을 참조하십시오.

## 클라이언트 만들기

코드에 사용할 리전으로 MediaConvert 클라이언트를 생성하여 AWS SDK for PHP를 구성합니다. 예를 들면, 리전이 us-west-2로 설정되어 있습니다. AWS Elemental MediaConvert는 계정별로 사용자 지정 엔드포인트를 사용하므로 `AWS.MediaConvert client class`도 계정별 엔드포인트를 사용하도록 구성합니다. 이렇게 하려면 엔드포인트 파라미터를 계정별 엔드포인트 (p. 192)로 설정합니다.

가져오기

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

```
use Aws\MediaConvert\MediaConvertClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default',
    'endpoint' => 'ACCOUNT_ENDPOINT'
]);
```

## 단순 트랜스코딩 작업 정의

트랜스코드 작업 파라미터를 정의하는 JSON을 생성합니다.

이러한 파라미터는 세부적으로 정의됩니다. AWS Elemental MediaConvert 콘솔에서 작업 설정을 선택한 다음, 작업 섹션 하단에 있는 작업 JSON 표시를 선택하여 JSON 작업 파라미터를 생성합니다. 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 object`를 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',
    'endpoint' => 'ACCOUNT_ENDPOINT'
]);

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',
    'endpoint' => 'ACCOUNT_ENDPOINT'
]);

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',
    'endpoint' => 'ACCOUNT_ENDPOINT'
]);

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 사용하는 Amazon S3 예제

Amazon Simple Storage Service(Amazon S3)는 확장성이 뛰어난 클라우드 스토리지를 제공하는 웹 서비스입니다. Amazon S3;는 간단한 웹 서비스 인터페이스를 통해 웹 어디서나 원하는 양의 데이터를 저장 및 검색할 수 있어 사용하기 편리한 객체 스토리지를 제공합니다.


AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

주제

- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 생성 및 사용 (p. 198)
- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 액세스 권한 관리 (p. 200)
- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 구성 (p. 202)
- AWS SDK for PHP 버전 3을 통해 Amazon S3 멀티파트 업로드 사용 (p. 204)
- AWS SDK for PHP 버전 3을 사용한 Amazon S3 미리 서명된 POST (p. 211)
- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 미리 서명된 URL (p. 212)
- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷을 정적 웹 호스트로 사용 (p. 214)
- AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 정책 작업 (p. 215)

# AWS SDK for PHP 버전 3을 사용하여 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

가져오기

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

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

## 버킷 목록 생성

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 만들기

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

```
$BUCKET_NAME = '<BUCKET-NAME>';

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

//Creating S3 Bucket
try {
```

```
    $result = $s3Client->createBucket([
        'Bucket' => $BUCKET_NAME,
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

## 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 사용하여 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 참조하십시오.

다음 예에서는 작업 방법을 보여줍니다.

- GetBucketAcl을 사용하여 버킷에 대한 액세스 제어 정책을 가져옵니다.
- PutBucketAcl을 사용하여 ACL로 버킷에 대한 권한을 설정합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 액세스 제어 목록 정책 가져오기 및 설정

가져오기

```
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 = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketAcl([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving bucket ACL as follows: \n";
    var_dump($resp);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Sets the permissions on a bucket using access control lists (ACL).
$params = [
    'ACL' => 'public-read',
    'AccessControlPolicy' => [
        // Information can be retrieved from `getBucketAcl` response
        'Grants' => [
            [
                'Grantee' => [
                    'DisplayName' => '<string>',
                    'EmailAddress' => '<string>',
                    'ID' => '<string>',
                    'Type' => 'CanonicalUser',
                    'URI' => '<string>',
                ],
                'Permission' => 'FULL_CONTROL',
            ],
```

```
            // ...
        ],
        'Owner' => [
            'DisplayName' => '<string>',
            'ID' => '<string>',
        ],
    ],
    'Bucket' => $bucket,
];

try {
    $resp = $s3Client->putBucketAcl($params);
    echo "Succeed in setting bucket ACL.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

# AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 구성

CORS(Cross-origin 리소스 공유)는 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. Amazon S3의 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 CORS(Cross-Origin Resource Sharing)를 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 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.

가져오기

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

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

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.

가져오기

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

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

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 3을 통해 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 객체를 Amazon S3에 업로드합니다.
- MultipartUploader를 사용하여 Amazon S3 객체에 멀티파트 업로드를 생성합니다.
- ObjectCopier를 사용하여 객체를 Amazon S3 로케이션끼리 복사합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 객체 업로더

`PutObject`와 `MultipartUploader` 중에서 어떤 것이 작업에 가장 적합한지 잘 모를 경우에는 `ObjectUploader`를 사용하십시오. `ObjectUploader`는 `PutObject`와 `MultipartUploader` 중에서 페이로드 크기를 기준으로 가장 적합한 것을 사용해 대용량 파일 하나를 Amazon S3에 업로드합니다.

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

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

샘플 코드

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

$bucket = 'your-bucket';
$key = 'my-file.zip';

// Using stream instead of 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);
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));
```

## 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.

가져오기

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

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

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.

가져오기

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

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

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 (\Aws\Command $command) {
        // $command is a CreateMultipartUpload operation
        $command['CacheControl'] = 'max-age=3600';
    },
    'before_upload' => function (\Aws\Command $command) {
        // $command is an UploadPart operation
        $command['RequestPayer'] = 'requester';
    },
    'before_complete' => function (\Aws\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 알고

리즘을 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.

가져오기

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

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

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);
}
```

`UploadState`에서 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 `catch` 블록 내부의 `$source` 변수를 재설정합니다.

가져오기

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

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

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));
```

## 멀티파트 업로드 중단

때로는 오류가 발생할 때 업로드를 다시 시작하지 않는 대신 전체를 중단해야 할 수 있습니다. 이 작업도 `UploadState` 객체에 포함된 데이터를 사용하여 쉽게 수행할 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 `MultipartUploader`를 호출하는 것은 차단 요청입니다. 비동기 콘텍스트에서 작업하는 경우 멀티파트 업로드에 대한 promise (p. 58)를 가져올 수 있습니다.

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

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

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.

## 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\S3\S3Client;
use Aws\Exception\AwsException;
use Aws\S3\MultipartCopy;
use Aws\Exception\MultipartUploadException;
```

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 S3 미리 서명된 POST

미리 서명된 URL과 마찬가지로 미리 서명된 POST를 사용하여 AWS 자격 증명을 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 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 S3 Developer Guide의 정책 생성 참조)
- 정책에 대한 만료 시간 문자열(선택 사항, 기본적으로 1시간)

가져오기

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

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

샘플 코드

```
$client = new S3Client([
    'profile' => 'default',
    'version' => 'latest',
    'region' => 'us-west-2',
]);
$bucket = 'mybucket';

// Set some 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', 'user/eric/'],
];

// Optional: configure expiration time string
$expires = '+2 hours';

$postObject = new \Aws\S3\PostObjectV4(
    $client,
    $bucket,
    $formInputs,
    $options,
    $expires
);

// Get attributes to set on an HTML form, e.g., action, method, enctype
$formAttributes = $postObject->getFormAttributes();

// Get form input fields. This will include anything set as a form input in
// the constructor, the provided JSON policy, your AWS access key ID, and an
// auth signature.
$formInputs = $postObject->getFormInputs();
```

# AWS SDK for PHP 버전 3을 사용하여 Amazon S3 미리 서명된 URL

HTTP 인증 헤더를 사용하는 대신 쿼리 문자열 파라미터로 필요한 정보를 전달하여 특정 유형의 요청을 인증할 수 있습니다. 이 방법은 요청을 프록시하지 않고 Amazon S3 프라이빗 데이터에 타사 브라우저가 액세스할 수 있도록 할 경우에 유용합니다. 방법은 "미리 서명된" 요청을 작성한 후 이를 최종 사용자의 브라우저가 검색할 수 있는 URL로 인코딩하는 것입니다. 또한 만료 시간을 지정하여 미리 서명된 요청을 제한할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- createPresignedRequest를 사용해 S3 객체를 가져올, 미리 서명된 URL을 생성합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 미리 서명된 요청 생성

`Aws\S3\S3Client::createPresignedRequest()` 메서드를 사용하여 Amazon S3 객체에 대한 미리 서명된 URL을 가져올 수 있습니다. 이 메서드는 `Aws\CommandInterface` 객체와 만료된 타임스탬프를 받아 미리 서명된 `Psr\Http\Message\RequestInterface` 객체를 반환합니다. 요청의 `getUri()` 메서드를 사용하여 객체의 미리 서명된 URL을 검색할 수 있습니다.

가장 일반적인 시나리오는 객체에 대해 GET을 실행하는 미리 서명된 URL을 생성하는 것입니다.

가져오기

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

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

샘플 코드

```
$s3Client = new Aws\S3\S3Client([
    'profile' => 'default',
    'region' => 'us-east-2',
    'version' => '2006-03-01',
]);

$cmd = $s3Client->getCommand('GetObject', [
    'Bucket' => 'my-bucket',
    'Key' => 'testKey'
]);

$request = $s3Client->createPresignedRequest($cmd, '+20 minutes');
```

## 미리 서명된 URL 생성

명령 객체를 생성하는 `getCommand` 메서드를 사용한 후 `createPresignedRequest()` 메서드를 호출하여 Amazon S3 작업에 대한 미리 서명된 URL을 생성할 수 있습니다. 요청을 전송할 때 반환되는 요청과 동일한 메서드와 동일한 헤더를 사용해야 합니다.

샘플 코드

```
//Creating a presigned URL
$cmd = $s3Client->getCommand('GetObject', [
    'Bucket' => 'my-bucket',
    'Key' => 'testKey'
]);

$request = $s3Client->createPresignedRequest($cmd, '+20 minutes');

// Get the actual presigned-url
$presignedUrl = (string)$request->getUri();
```

## 객체에 대한 URL 가져오기

Amazon S3 버킷에 저장된 객체에 대한 퍼블릭 URL만 필요한 경우 `Aws\S3\S3Client::getObjectUrl()` 메서드를 사용할 수 있습니다. 이 메서드는 지정된 버킷 및 키에 대한 부호 없는 URL을 반환합니다.

샘플 코드

```
//Getting the URL to an object
$url = $s3Client->getObjectUrl('my-bucket', 'my-key');
```

> **Important**
>
> 이 메서드에 의해 반환되는 URL은 버킷 또는 키가 존재하는지 검증되지 않았으며, 이 메서드는 객체가 무단 액세스를 허용하는지 여부를 확인하지 않습니다.

# AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷을 정적 웹 호스트로 사용

Amazon S3에 정적 웹 사이트를 호스팅할 수 있습니다. 자세한 내용은 S3 정적 웹 사이트 호스팅을 참조하십시오.

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.

## 자격 증명

예제 코드를 실행하기 전에 AWS 자격 증명을 구성합니다. AWD SDK for PHP 버전 3의 자격 증명 (p. 39)을 참조하십시오.

## 버킷에 대한 웹 사이트 구성 가져오기, 설정, 삭제

가져오기

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

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

샘플 코드

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

// Retrieving the Bucket Website Configuration
$bucket = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
        ],
```

```
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ]
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

# AWS SDK for PHP 버전 3을 사용하여 Amazon S3 버킷 정책 작업

버킷 정책을 사용하여 Amazon S3 리소스에 대한 권한을 부여할 수 있습니다. 자세한 내용은 버킷 정책 및 사용자 정책 사용을 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 버킷에 대한 정책 가져오기, 삭제, 바꾸기

가져오기

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

use Aws\S3\S3Client;
```

```
use Aws\Exception\AwsException;
```

샘플 코드

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

$bucket = 'my-s3-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
    echo "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deletes the policy from the bucket
try {
    $resp = $s3Client->deleteBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Replaces a policy on the bucket
try {
    $resp = $s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => 'foo policy',
    ]);
    echo "Succeed in put a policy on bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

# Secrets Manager API 및 AWS SDK for PHP 버전 3을 사용하여 비밀 관리

AWS Secrets Manager는 암호, API 키, 데이터베이스 자격 증명 같은 공유 비밀을 저장하고 관리합니다. 개발자는 이러한 Secrets Manager 서비스에서 Secrets Manager 호출 기능을 사용해 배포 코드에서 하드 코딩된 자격 증명을 변경할 수 있습니다.

Secrets Manager는 Amazon Relational Database Service(Amazon RDS) 데이터베이스를 대상으로 자동으로 예약되는 자격 증명 교체 기능을 기본적으로 지원하여 애플리케이션 보안을 강화합니다. 그 밖에도 Secrets Manager는 AWS Lambda를 사용해 서비스별 세부 정보를 구현함으로써 다른 데이터베이스나 타사 서비스의 비밀까지도 원활하게 교체할 수 있습니다.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- CreateSecret을 사용해 비밀을 생성합니다.
- GetSecretValue를 사용해 비밀을 가져옵니다.
- ListSecrets를 사용해 Secrets Manager에서 저장된 비밀을 모두 나열합니다.
- DescribeSecret을 사용해 특정 비밀에 대한 세부 정보를 가져옵니다.
- PutSecretValue를 사용해 특정 비밀을 업데이트합니다.
- RotateSecret을 사용해 비밀 교체를 설정합니다.
- DeleteSecret을 사용해 삭제할 비밀을 표시합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

# 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

# Secrets Manager에서 비밀 생성

Secrets Manager에서 비밀을 생성할 때는 CreateSecret 작업을 사용합니다.

이번 예제에서는 사용자 이름과 암호가 JSON 문자열로 저장되어 있습니다.

가져오기

```
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 = '{"username":"<<USERNAME>>","password":"<<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 비밀 가져오기

Secrets Manager에 저장된 비밀 값을 가져올 때는 GetSecretValue 작업을 사용합니다.

이번 예제에서는 비밀이 저장된 값이 포함된 문자열입니다. 앞에서 생성한 비밀과 관련하여 호출할 경우 이번 샘플 코드에서는 `[{\"username\":\"<<USERNAME>>\",\"password\":\"<<PASSWORD>>\"}]`가 출력됩니다. `json.loads`를 사용하여 인덱싱 값에 액세스합니다.

가져오기

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => '<<{{MyRegionName}}>>',
]);

$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']);
}

// Your code goes here;
```

# Secrets Manager에 저장된 비밀 나열

Secrets Manager에서 저장된 비밀 목록을 모두 나열할 때는 ListSecrets 작업을 사용합니다.

가져오기

```
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";
}
```

# 비밀에 대한 세부 정보 가져오기

저장된 비밀에는 교체 규칙에 대한 메타데이터, 마지막 액세스 또는 변경 시점, 사용자 생성 태그, Amazon 리소스 이름(ARN)이 포함되어 있습니다. Secrets Manager에 저장된 특정 비밀에 대한 세부 정보를 가져올 때는 DescribeSecret 작업을 사용합니다.

가져오기

```
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";
}
```

# 비밀 값 업데이트

새롭게 암호화된 비밀 값을 Secrets Manager에 저장할 때는PutSecretValue 작업을 사용합니다.

그러면 새로운 버전의 비밀이 생성됩니다. 비밀 버전이 이미 존재한다면 값과 함께 `VersionStages` 파라미터를 `AWSCURRENT`에 추가하여 값을 가져올 때 새로운 값이 사용되도록 합니다.

가져오기

```
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 = '{"username":"<<USERNAME>>","password":"<<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 비밀 값 교체

Secrets Manager에 저장된 기존 비밀 값을 교체할 때는 Lambda 교체 함수와 RotateSecret 작업을 사용합니다.

시작하기 전에 먼저 비밀을 교체할 때 사용할 Lambda 함수를 생성합니다. 현재 AWS 코드 샘플 카탈로그에는 Amazon RDS 데이터베이스 자격 증명을 교체할 때 사용되는 Lambda 코드 예제가 몇 가지 포함되어 있습니다.

> **Note**
>
> 비밀 교체에 대한 자세한 내용은 AWS Secrets Manager User Guide에서 AWS Secrets Manager 비밀 교체 단원을 참조하십시오.

Lambda 함수를 설정하였으면 이제 새로운 비밀 교체를 구성합니다.

가져오기

```
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.

가져오기

```
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 비밀 삭제

Secrets Manager에서 지정된 비밀을 제거할 때는 DeleteSecret 작업을 사용합니다. 비밀을 우발적으로 삭제하지 못하도록 DeletionDate 스탬프가 비밀에 자동으로 추가되어 삭제를 되돌릴 수 있는 복구 시간을 지정할 수 잇습니다. 복구 시간을 지정하지 않으면 기본 시간으로 30일이 지정됩니다.

가져오기

```
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

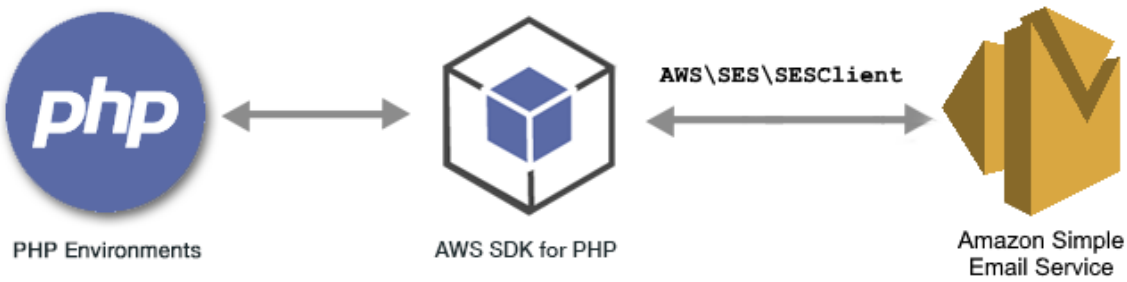
AWS SDK for PHP 예제는 AWS Secrets Manager API Reference에서 다음 REST 작업을 사용합니다.

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

AWS Secrets Manager 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서 단원을 참조하십시오.

# AWS SDK for PHP 버전 3을 사용하는 Amazon SES 예제

Amazon Simple Email Service(Amazon SES)는 사용자의 이메일 주소와 도메인을 사용해 이메일을 보내고 받기 위한, 비용을 절감할 수 있는 손쉬운 방법을 제공하는 이메일 플랫폼입니다. Amazon SES에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.




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 자격 증명 확인 (p. 224)
- Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 사용자 지정 이메일 템플릿 만들기 (p. 228)
- Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 필터 관리 (p. 232)
- Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 규칙 생성 및 관리 (p. 235)
- Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 전송 활동 모니터링 (p. 241)
- Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 발신자에게 권한 부여 (p. 242)

# Amazon SES API 및 AWS SDK for PHP 버전 3을 사용하여 이메일 자격 증명 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 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 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.

가져오기

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

use Aws\SES\SESClient;
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 SES Developer Guide의 Amazon SES로 도메인 확인을 참조하십시오.

Amazon SES 계정에 전송 도메인을 추가하려면 VerifyDomainIdentity 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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

자격 증명 목록에서 확인된 이메일 주소를 삭제하려면 DeleteIdentity 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
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 자격 증명 목록에서 확인된 이메일 도메인을 삭제하려면 DeleteIdentity 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
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 만들기

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 SES Developer Guide의 Sending Personalized Email Using the Amazon SES 단원을 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 이메일 템플릿 만들기

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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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 계정과 연결된 모든 이메일 템플릿의 목록을 검색하려면 ListTemplates 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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' => $verified_recipient_emails,
        ],
        'ReplyToAddresses' => [$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 전송하는 것 외에도 Amazon Simple Email Service(Amazon SES)로 이메일을 수신할 수도 있습니다. IP 주소 필터는 특정 IP 주소 또는 특정 범위의 IP 주소에서 발신한 메일의 수락 여부를 지정할 수 있게 합니다. 자세한 내용은 Managing IP Address Filters for Amazon SES Email Receiving 단원을 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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 전송하는 것 외에도 Amazon Simple Email Service(Amazon SES)로 이메일을 수신할 수도 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 또는 AWS Lambda를 포함하지만 이에 국한되지 않는 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 참조하십시오.

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 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 수신 규칙 세트 만들기

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.

가져오기

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

use Aws\SES\SESClient;
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 만들기

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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

```
use Aws\SES\SESClient;
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 아래에 있는 수신 규칙 세트를 나열하려면 ListReceiptRuleSets 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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.

가져오기

```
```

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

use Aws\SES\SESClient;
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.

가져오기

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

use Aws\SES\SESClient;
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 전송 활동 모니터링

Amazon Simple Email Service(Amazon SES)는 전송 활동을 모니터링하기 위한 방법을 제공합니다. 이러한 방법을 사용하여 계정의 반송, 수신 거부 및 거부 메일의 비율 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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 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 참조하십시오.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

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 발신자에게 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 자격 증명에 연결하는 JSON 문서입니다.

이 정책은 누가 어떤 조건으로 해당 자격 증명을 사용하여 전송할 수 있는지 명시적으로 나열합니다. 사용자와 사용자가 정책에서 명시적으로 권한을 부여한 개체 이외의 모든 발신자는 이메일을 전송할 수 없습니다. 자격 증명은 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 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

Amazon SES 사용에 대한 자세한 내용은 Amazon SES 개발자 안내서를 참조하십시오.

## 권한 있는 발신자 만들기

다른 AWS 계정에 사용자를 대신해 이메일을 전송할 수 있는 권한을 부여하려면 자격 증명 정책을 사용하거나 확인된 이메일 주소 또는 도메인에서 이메일을 전송하도록 권한 부여를 추가하거나 업데이트합니다. 자격 증명 정책을 만들려면 PutIdentityPolicy 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

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 자격 증명 또는 도메인 자격 증명과 연결된 전송 권한 부여 정책을 반환합니다. 지정된 이메일 주소 또는 도메인에 대한 전송 권한 부여를 가져오려면 GetIdentityPolicy 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

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 AWS 리전의 특정 이메일 자격 증명 또는 도메인 자격 증명과 연결된 전송 권한 부여 정책을 나열하려면 ListIdentityPolicies 작업을 사용합니다.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

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 자격 증명 정책을 삭제하여 이메일 자격 증명 또는 도메인 자격 증명으로 이메일을 전송할 수 있는 다른 AWS 계정의 전송 권한 부여를 제거합니다.

가져오기

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

use Aws\SES\SESClient;
use Aws\Exception\AwsException;
```

샘플 코드

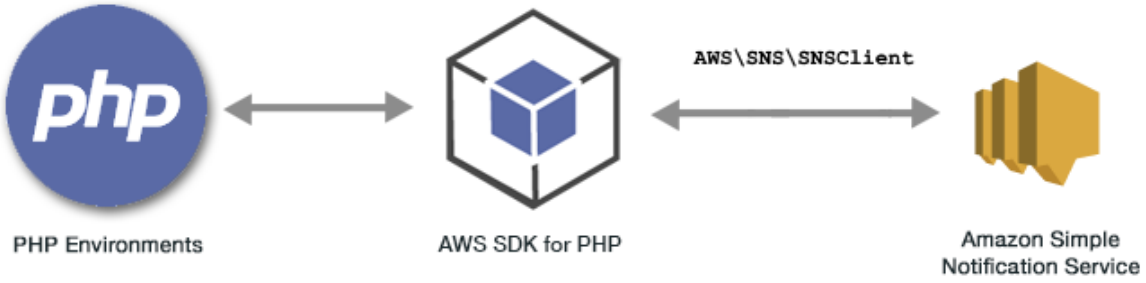
```
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-1'
]);

$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 사용하는 Amazon SNS 예제

Amazon Simple Notification Service(Amazon SNS)는 구독 중인 endpoint 또는 클라이언트에 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 3으로 Amazon SNS 주제 관리 (p. 246)
- AWS SDK for PHP 버전 3으로 Amazon SNS에서 구독 관리 (p. 250)
- AWS SDK for PHP 버전 3으로 Amazon SNS에서 SMS 메시지 전송 (p. 255)

## AWS SDK for PHP 버전 3으로 Amazon SNS 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.

PHP용 AWS SDK 버전 3으로 Amazon SNS에서 구독 관리 (p. 250)에서 주제 구독에 대해 자세히 알아보십시오.

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 참조하십시오.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 주제 생성

주제를 생성하려면 CreateTopic 작업을 사용합니다.

AWS 계정의 각 주제 이름은 고유해야 합니다.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

```
use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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 3으로 Amazon SNS에서 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 Lambda에 알림을 전송합니다.

구독은 구독자에 대한 메시지 전송을 관리하는 주제에 연결되어 있습니다. 주제 생성에 대한 자세한 내용은 PHP용 AWS SDK 버전 3으로 Amazon SNS 주제 관리 (p. 246) 단원을 참조하십시오.

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 Using Amazon SNS for System-to-System Messaging 단원을 참조하십시오.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 이메일 주소가 주제를 구독하도록 등록

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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 함수가 주제를 구독하도록 등록

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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 PHP용 AWS SDK 버전 3으로 Amazon SNS에서 SMS 메시지 전송 (p. 255)을 참조하십시오.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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->subscribe([
        '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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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 3으로 Amazon SNS에서 SMS 메시지 전송

Amazon Simple Notification Service(Amazon SNS)를 사용하여 SMS 수신 가능한 디바이스에 텍스트 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 참조하십시오.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

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.

가져오기

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

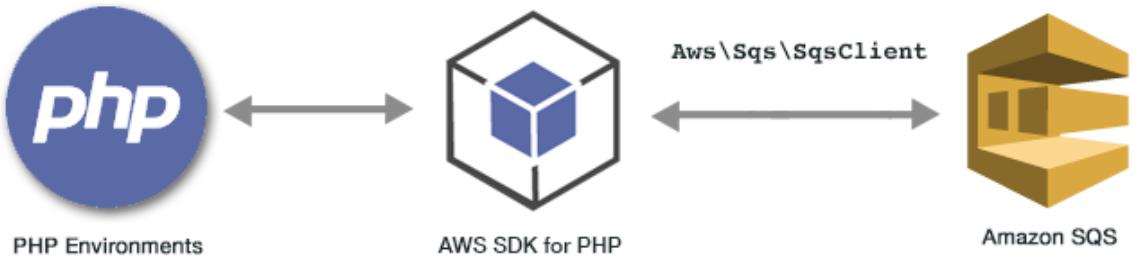
샘플 코드

```
$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 사용하는 Amazon SQS 예제

Amazon Simple Queue Service(SQS)는 빠르고, 안정적이고, 확장 가능한 완전 관리형 메시지 대기열 서비스입니다. Amazon SQS를 사용하면 클라우드 애플리케이션의 구성 요소를 분리할 수 있습니다. Amazon SQS에는 처리량이 많고 최소 1회 처리되는 표준 대기열과 FIFO(선입선출) 전송 및 정확히 1회 처리를 제공하는 FIFO 대기열이 포함됩니다.




AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

주제

- AWS SDK for PHP 버전 3으로 Amazon SQS에서 긴 폴링 활성화 (p. 259)
- AWS SDK for PHP 버전 3으로 Amazon SQS에서 제한 시간 초과 관리 (p. 262)
- AWS SDK for PHP 버전 3으로 Amazon SQS 메시지 전송 및 수신 (p. 263)
- AWS SDK for PHP 버전 3으로 Amazon SQS에서 배달 못한 편지 대기열 사용 (p. 265)
- PHP용 AWS SDK 버전 3으로 Amazon SQS 멀티파트 업로드 사용 (p. 266)

## AWS SDK for PHP 버전 3으로 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 참조하십시오.

다음 예제에서는 다음과 같은 작업을 하는 방법을 보여줍니다.

- SetQueueAttributes를 사용하여 Amazon SQS 대기열에서 긴 폴링을 활성화하는 속성을 설정합니다.
- ReceiveMessage를 사용하여 긴 폴링이 있는 하나 이상의 메시지를 검색합니다.
- CreateQueue를 사용하여 긴 폴링 대기열을 생성합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 긴 폴링을 활성화하기 위한 대기열의 속성 설정

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$queueUrl = "QUEUE_URL";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes(array(
        '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

가져오기

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

```
use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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' => 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

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$queueName = "QUEUE_NAME";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue(array(
        'QueueName' => $queueName,
        'Attributes' => array(
            'ReceiveMessageWaitTimeSeconds' => 20
        ),
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

# AWS SDK for PHP 버전 3으로 Amazon SQS에서 제한 시간 초과 관리

제한 시간 초과는 Amazon SQS에서 다른 사용 구성 요소가 메시지를 수신하고 처리할 수 없는 기간입니다. 자세히 알아보려면 제한 시간 초과를 참조하십시오.

다음 예에서는 작업 방법을 보여줍니다.

- ChangeMessageVisibilityBatch를 사용하여 대기열에서 지정된 메시지의 제한 시간 초과를 새 값으로 변경합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 여러 메시지의 제한 시간 초과 변경

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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); $i++) {
            array_push($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 3으로 Amazon SQS 메시지 전송 및 수신

Amazon SQS 메시지에 대한 자세한 내용은 Amazon SQS Developer Guide의 SQS 대기열에 메시지 보내기 및 SQS 대기열에서 메시지 받기 및 삭제를 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 메시지 보내기

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ));
    if (count($result->get('Messages')) > 0) {
        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 3으로 Amazon SQS에서 배달 못한 편지 대기열 사용

배달 못한 편지 대기열은 다른 (소스) 대기열에서 성공적으로 처리하지 못한 메시지를 보낼 수 있는 대기열입니다. 배달 못한 편지 대기열에서 이 메시지를 구분하고 격리하여 처리에 실패한 이유를 확인할 수 있습니다. 배달 못한 편지 대기열로 메시지를 보내는 각 소스 대기열을 개별적으로 구성해야 합니다. 여러 대기열에서 하나의 배달 못한 편지 대기열로 메시지를 보낼 수 있습니다.

자세한 내용은 SQS 배달 못한 편지 대기열 사용을 참조하십시오.

다음 예에서는 작업 방법을 보여줍니다.

- SetQueueAttributes를 사용하여 배달 못한 편지 대기열을 활성화합니다.

AWS SDK for PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 배달 못한 편지 대기열 활성화

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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());
}
```

# PHP용 AWS SDK 버전 3으로 Amazon SQS 멀티파트 업로드 사용

Amazon SQS 대기열에 대한 자세한 내용은 SQS 대기열의 작동 방식을 참조하십시오.

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 PHP 버전 3에 대한 모든 예제 코드는 GitHub에서 사용할 수 있습니다.

## 자격 증명

예제 코드를 실행하기 전에 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)에 설명된 대로 AWS 자격 증명을 구성합니다. PHP용 AWS SDK 버전 3의 기본 사용 패턴 (p. 7)에 설명된 대로 AWS SDK for PHP를 가져옵니다.

## 대기열 목록 반환

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

샘플 코드

```
$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue(array(
        'QueueName' => $queueName,
        'Attributes' => array(
            'DelaySeconds' => 5,
            'MaximumMessageSize' => 4096, // 4 KB
        ),
    ));
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

## 대기열의 URL 반환

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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

가져오기

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

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 3 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 경우를 업데이트해야 할 수도 있습니다.

클라이언트의 API 공급자가 찾을 수 있는 최신 API 버전을 사용하려면 `latest` 문자열을 "version" 구성 값에 제공할 수 있습니다. 기본 api_provider는 API용 SDK의 `src/data` 디렉터리를 스캔합니다.

> 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. 사용 가능한 AWS 리전과 엔드포인트의 목록은 Amazon Web Services General Reference의 AWS 리전 및 엔드포인트를 참조하십시오.

```
// 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 참조하십시오.

# 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 `true`로 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\CommandInterface`의 `Aws\ClientInterface`에 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 mapRequest (p. 67)를 참조하십시오.

# 임의 요청에 서명하려면 어떻게 해야 하나요?

SDK의 :aws-php-class: SignatureV4 클래스</class-Aws.Signature.SignatureV4.html>를 사용하여 임의의 :aws-php-class: PSR-7 요청</class-Psr.Http.Message.RequestInterface.html>에 서명할 수 있습니다.

이렇게 수행하는 방법에 대한 전체 예제는 PHP용 AWS SDK 버전 3으로 사용자 지정 Amazon CloudSearch 도메인 요청에 서명 (p. 122)을 참조하십시오.

# 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 mapCommand (p. 66)를 참조하십시오.

# CredentialsException이란 무엇입니까?

AWS SDK for PHP를 사용하는 동안 `Aws\Exception\CredentialsException`이 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 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 PHP용 AWS SDK 버전 3의 자격 증명 (p. 39)을 참조하십시오.

# 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. 리전, 엔드포인트, 지원되는 스키마 목록은 Amazon Web Services General Reference의 AWS 리전 및 엔드포인트를 참조하십시오.

```
$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 PHP용 AWS SDK 버전 3에 대한 요구 사항 및 권장 사항 (p. 4)을 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 (p. 31) 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_content (p. 31)를 참조하십시오.

# 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 는 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 `3`으로 구성되며, Amazon DynamoDB만 예외적으로 `10`으로 구성됩니다.

# 오류 코드가 있는 예외를 처리하려면 어떻게 해야 하나요?

AWS SDK for PHP-customized `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`의 getAwsErrorCode()에서 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 지정 (p. 38)합니다.

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 객체를 생성 및 사용하는 방법에 대한 자세한 내용은 기본 사용 설명서 (p. 7)를 참조하십시오.

명령

명령 객체는 작업 실행을 캡슐화합니다. SDK의 기본 사용 패턴 (p. 7)을 따를 때 명령 객체를 직접 처리하지 않습니다. 동시 요청, 일괄 처리 등과 같은 SDK의 고급 기능을 사용하려면 클라이언트의 `getCommand()` 메서드를 사용하여 명령 객체에 액세스할 수 있습니다. 자세한 내용은 PHP용 AWS SDK 버전 3의 명령 객체 (p. 53) 가이드를 참조하십시오.

자격 증명

AWS 서비스와 상호 작용하려면 자격 증명 또는 AWS 액세스 키를 사용하여 서비스에 인증합니다. 액세스 키는 계정을 식별하는 액세스 키 ID와 작업을 실행할 때 서명을 생성하는 데 사용되는 보안 액세스의 두 부분으로 구성됩니다. 클라이언트 객체를 구성할 때 자격 증명을 제공 (p. 39)합니다.

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. 는 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 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.

작업

서비스 API의 단일 작업(예: DynamoDB의 `CreateTable`, Amazon EC2의 `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 지

정 API를 더 쉽게 사용할 수 있도록 이 프로세스의 추상화 역할을 하는 SDK 기능입니다. 이 기능은 클라이언트의 `getPaginator()` 메서드를 통해 액세스합니다. 자세한 내용은 PHP용 AWS SDK 버전 3의 페이지네이터 (p. 73) 가이드를 참조하십시오.

Promise

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.

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 제공 (p. 36)합니다.

SDK

"SDK"라는 용어는 전체적으로 AWS SDK for PHP 라이브러리를 의미하지만, 각 서비스에 대해 클라이언트 객체의 팩토리 역할을 하는 `Aws\Sdk` 클래스(docs)를 의미할 수도 있습니다. 또한 `Sdk` 클래스를 사용하여 생성되는 모든 클라이언트 객체에 적용되는 글로벌 구성 값 (p. 23) 세트를 제공할 수 있습니다.

서비스

AWS 서비스를 나타내는 일반적인 방법입니다(예: Amazon S3, Amazon DynamoDB, AWS OpsWorks). 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.

서명

작업을 실행할 때 SDK에서는 자격 증명을 사용하여 요청에 대한 디지털 서명을 생성합니다. 그러면 서비스에서 요청을 처리하기 전에 서명을 확인합니다. 서명 프로세스는 SDK에 의해 캡슐화되며 클라이언트에 대해 구성된 자격 증명을 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 PHP용 AWS SDK 버전 3의 Waiter (p. 75) 가이드를 참조하십시오.

최신 AWS 용어는 AWS 일반 참조의 AWS 용어집을 참조하십시오.

# 추가 리소스

다음 링크는 PHP용 SDK 버전 3에서 사용할 수 있는 추가 리소스를 제공합니다.

## PHP SDK 포럼

PHP SDK 포럼에서 PHP용 SDK 사용자의 관심사에 대한 질문 및 토론을 찾아 볼 수 있습니다.

## GitHub의 PHP SDK 버전 3 및 개발자 가이드

GitHub에는 AWS SDK for PHP용으로 사용할 수 있는 여러 리포지토리가 있습니다.

- 최신 AWS SDK for PHP는 SDK 리포지토리에서 구할 수 있습니다.
- PHP용 SDK 개발자 안내서는 자체 문서 리포지토리에 재구성된 텍스트 형식으로 제공됩니다.
- 이 가이드에 포함된 샘플 코드는 SDK 샘플 코드 리포지토리에서 사용할 수 있습니다.

## Gitter의 PHP SDK

또한 Gitter의 PHP SDK 커뮤니티에서 AWS SDK for PHP에 대한 질문 및 토론을 찾아 볼 수 있습니다.

# 문서 이력

다음 표에서는 AWS SDK for PHP Developer Guide의 최신 릴리스 이후 변경된 중요 사항에 대해 설명합니다.

설명서 최종 메이저 업데이트: 2018년 5월 9일

| 변경 사항 | 설명 | 릴리스 날짜 |
|---|---|---|
| Secrets Manager 예 | 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 |
| Amazon SES 예 | 다른 서비스 예제 추가 | 2018년 10월 5일 |
| AWS KMS 예 | 다른 서비스 예제 추가 | 2018년 8월 8일 |
| 자격 증명 | 자격 증명 가이드 설명 및 간소화 | 2018년 6월 30일 |
| MediaConvert 예 | 다른 서비스 예제 추가 | 2018년 6월 15일 |
| 새 웹 레이아웃 | 설명서를 AWS 스타일로 전환 | 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 |
| 자격 증명 추가 | AssumeRole 및 ini에 대한 지원 추가 | 2017년 1월 17일 |
| S3 예제 | S3 다중 리전 및 미리 서명된 게시물 | 2016년 3월 18일 |
| Amazon ES 및 Amazon CloudSearch | 다른 서비스 예제 추가 | 2015년 12월 28일 |
| 명령줄 | 명령 파라미터 추가 | 2015년 8월 13일 |
| 서비스 기능 | S3 및 AWS에 대한 서비스 기능 추가 | 2015년 4월 30일 |
| 새 SDK 버전 | AWS SDK for PHP의 버전 3이 출시되었습니다. | 2015년 26월 5일 |