
AWS SDK para PHP

Guía para desarrolladores

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

AWS SDK para PHP	1
Introducción	1
Guías de SDK	2
Características específicas de servicios	2
Examples	2
Reference	2
Documentación de la API	3
Mantenimiento y compatibilidad con las versiones principales del SDK	3
Introducción	4
Requisitos previos	4
Requirements	4
Recommendations	4
Prueba de compatibilidad	5
Instalación del SDK de	5
Instalar AWS SDK for PHP como una dependencia mediante Composer	5
Instalación con el phar empaquetado	6
Instalación con el archivo ZIP	7
Uso básico	7
Prerequisites	7
Inclusión del SDK en su código	7
Resumen de uso	7
Crear un cliente	8
Usar la clase Sdk	8
Ejecutar operaciones de servicio	9
Solicitudes asíncronas	10
Uso de objetos Result	11
Gestión de errores	12
Actualización desde la versión 2	14
Introduction	14
¿Qué novedades incluye la versión 3?	14
¿Cuáles son las diferencias con la versión 2?	15
Comparación de ejemplos de código de ambas versiones del SDK	21
Configuración del SDK	24
Opciones de configuración	24
api_provider	25
credentials	25
debug	27
stats	28
endpoint	29
endpoint_provider	29
endpoint_discovery	30
handler	31
http	32
http_handler	37
profile	38
region	39
retries	39
scheme	41
service	41
signature_provider	42
signature_version	42
ua_append	42
use_aws_shared_config_files (usar_archivos_de_configuración_compartidos_de_aws)	43
validate	43

version	44
Configuración de las credenciales de	44
Uso de la cadena predeterminada de proveedores de credenciales	45
Otras formas de añadir credenciales	45
Uso de credenciales desde las variables de entorno	46
Uso del archivo de credenciales de AWS y de los perfiles de credenciales	46
Cómo asumir un rol de IAM	48
Uso de un proveedor de credenciales	52
Uso de credenciales temporales desde AWS STS	59
Credenciales codificadas	60
Creación de clientes anónimos	61
Objetos comando	61
Uso implícito de comandos	61
Parámetros de comando	62
Crear objetos comando	63
Comando HandlerList	63
CommandPool	64
Promesas	66
¿Qué es una promesa?	67
Promesas en el SDK	67
Encadenar promesas	68
Esperar a las promesas	69
Cancelar promesas	70
Combinar promesas	70
Controladores y middleware	71
Handlers	71
Middleware	73
Crear controladores personalizados	78
flujos	79
Decoradores de flujo	79
Paginadores	82
Objetos de paginador	82
Enumeración de datos a partir de los resultados	83
Paginación asíncrona	83
Esperadores	84
Configuración del esperador	84
Espera asíncrona	85
ExpresionesJMESPath	86
Extraer datos de los resultados	87
Extraer datos de paginadores	90
SDK Metrics	90
Autorizar las métricas del SDK	91
Configuración de SDK Metrics	93
Definiciones de métricas del SDK	97
Uso de los servicios de AWS	100
AWS Cloud9	100
Paso 1: Configurar su cuenta de AWS para usarAWS Cloud9	100
Paso 2: Configuración del entorno de desarrollo deAWS Cloud9	101
Paso 3: Configuración de la AWS SDK for PHP de	101
Paso 4: Descargar código de ejemplo	101
Paso 5: Ejecución y depuración del código de ejemplo	102
Amazon DynamoDB	102
Uso básico	102
Configuration	103
Pricing	104
Bloqueo de sesiones	105
Recopilación de elementos no utilizados	105

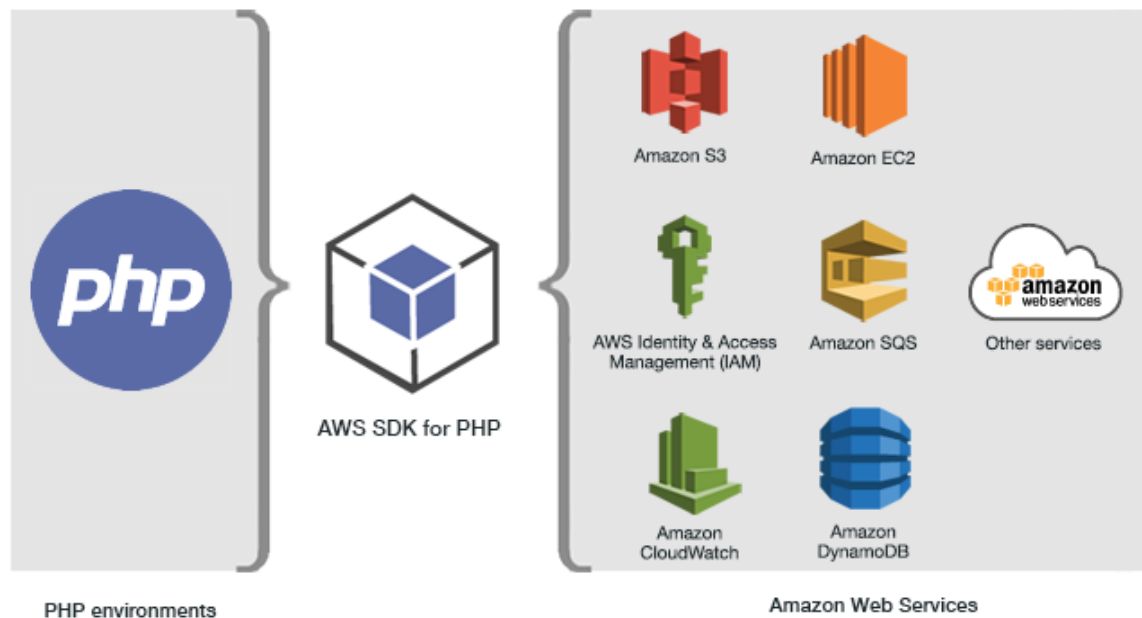
Prácticas recomendadas de	106
Permisos de IAM necesarios	107
Cliente de Amazon S3 para varias regiones	107
Uso básico	102
Caché de la región del bucket	108
Encapsulador de flujo de Amazon S3	108
Descargar datos	109
Cargar datos	110
Modos fopen	110
Otras funciones de objeto	111
Uso de buckets de	112
Opciones de contexto del flujo	96
Gestor de transferencias de Amazon S3	113
Carga de un directorio local en Amazon S3	113
Descarga de un bucket de Amazon S3	114
Configuration	103
Opciones de transferencia	115
Transferencias asíncronas	115
Personalización de los comandos del administrador de transferencias	116
Cifrado del lado del cliente de Amazon S3	116
Migration Guide	117
Setup	117
Encryption	117
Decryption	118
Configuración de Cipher	119
Estrategias de metadatos	119
Cargas multiparte	120
Ejemplos de código	122
Credentials	25
Ejemplos de AmazonCloudFront	122
Administración de distribuciones deCloudFront	123
Administración de invalidaciones deCloudFront	133
Firma de CloudFront URLs	137
Amazon CloudSearch	142
Credentials	25
Firma de una solicitud de dominio deCSlong	143
Ejemplos de AmazonCloudWatch	144
Credentials	25
Uso de alarmas de AmazonCloudWatch	145
Obtención de métricas de AmazonCloudWatch	148
Publicación de métricas personalizadas en AmazonCloudWatch	153
Envío de eventos a eventos de AmazonCloudWatch	156
Uso de acciones de alarma con alarmas de AmazonCloudWatch	159
Ejemplos de Amazon EC2	161
Credentials	25
Administración de instancias Amazon EC2	161
Uso de direcciones IP elásticas	164
Uso de regiones y zonas de disponibilidad	166
Uso de pares de claves de	167
Uso de grupos de seguridad	169
Amazon Elasticsearch	171
Credentials	25
Firma de una solicitud de Amazon ES	171
Ejemplos de AWS Identity and Access Management (IAM)	172
Credentials	25
Administración de las claves de acceso de IAM	173
Administración de usuarios de IAM	176

Uso de alias de cuenta de IAM	179
Uso de políticas de IAM	181
Uso de certificados de servidor de IAM	188
AWS Key Management Service	190
Uso de claves	191
Cifrado y descifrado de claves de datos	195
Uso de las políticas de claves	197
Uso de concesiones	200
Uso de alias	204
Ejemplos de Amazon Kinesis	207
Kinesis Data Streams	207
Fragmentos de Kinesis	211
Secuencias de entrega de Kinesis Data Firehose	213
AWS ElementalMediaConvert	219
Credentials	25
Obtención del punto de enlace específico de la cuenta	220
Creación y administración de trabajos	222
Ejemplos de Amazon S3	227
Credentials	25
Creación y uso de buckets de Amazon S3	227
Administración de permisos de acceso a los buckets de Amazon S3	230
Configuración de buckets de Amazon S3	231
Cargas multiparte de Amazon S3	233
Creación de POSTs con firma previa de S3	240
URL prefirmada de Amazon S3	241
Uso de un bucket de Amazon S3 como un host web estático	243
Uso de políticas de bucket de Amazon S3	244
Uso de ARNs de punto de acceso de S3	246
Amazon Secrets Manager	248
Credentials	25
Creación de un secreto en Secrets Manager	248
Recuperación de un secreto de Secrets Manager	249
Listado de los secretos almacenados en Secrets Manager	250
Recuperación de los detalles de un secreto	251
Actualización del valor del secreto	251
Rotación del valor de un secreto existente en Secrets Manager	252
Eliminación de un secreto de Secrets Manager	253
Información relacionada	254
Ejemplos de Amazon SES	254
Verificación de direcciones de correo electrónico	255
Uso de plantillas de correo electrónico	260
Administración de filtros de correo electrónico	264
Uso de reglas de correo electrónico	267
Monitorización de la actividad de envío	273
Autorización de remitentes	275
Ejemplos de Amazon SNS	278
Administración de temas	279
Administración de suscripciones	282
Envío de mensajes SMS en Amazon	288
Ejemplos de Amazon SQS	292
Habilitación del sondeo largo	292
Administración del tiempo de espera de visibilidad	295
Envío y recepción de mensajes	296
Uso de colas de mensajes fallidos	298
Uso de colas	299
Seguridad	302
Protección de los datos	302

Identity and Access Management	303
Validación de conformidad	303
Resistencia	304
Seguridad de la infraestructura	305
Migración del cliente de cifrado de S3	305
Información general sobre la migración	305
Actualizar clientes existentes para leer nuevos formatos	305
Migración de clientes de cifrado y descifrado a V2	306
Ejemplos de migración	306
PREGUNTAS FRECUENTES	309
¿Qué métodos están disponibles en un cliente?	309
¿Qué debo hacer si aparece un error de certificado SSL de cURL?	309
¿Qué versiones de la API están disponibles para un cliente?	309
¿Qué versiones de Region están disponibles para un cliente?	309
¿Por qué no puedo cargar o descargar archivos de más de 2 GB?	310
¿Cómo puedo ver qué datos se envían a través de la red?	310
¿Cómo puedo establecer los encabezados arbitrarios en una solicitud?	310
¿Cómo puedo firmar una solicitud arbitraria?	311
¿Cómo puedo modificar un comando antes de enviarlo?	311
¿Qué es un CredentialsException?	311
¿AWS SDK for PHP se puede utilizar en HHVM?	311
¿Cómo se deshabilita SSL?	311
¿Qué debo hacer cuando aparece un "Error de análisis"?	312
¿Por qué el cliente de Amazon S3 descomprime los archivos gz?	312
¿Cómo puedo deshabilitar la firma de cuerpos en Amazon S3?	312
¿Cómo se gestiona el esquema de reintento en AWS SDK for PHP?	313
¿Cómo se gestionan las excepciones con códigos de error?	313
Glossary	315
Recursos de adicionales	317
Foro del SDK para PHP	317
PHP SDK v3 y guía para desarrolladores enGitHub	317
SDK para PHP en Gitter	317
Historial del documento	318
.....	CCXX

¿Qué es el AWS SDK for PHP versión 3?

AWS SDK for PHP versión 3 permite a los desarrolladores de PHP utilizar [Amazon Web Services](#) en su código PHP y crear aplicaciones y software sólidos utilizando servicios como Amazon S3, Amazon DynamoDB, S3 Glacier, etc. Puede empezar rápidamente instalando el SDK mediante Composer (solicitando el paquete `aws/aws-sdk-php`) o descargando el archivo [aws.zip](#) o [aws.phar](#) independiente.



No todos los servicios están disponibles de forma inmediata en el SDK. Para saber qué servicios son compatibles actualmente con AWS SDK for PHP, consulte la sección sobre [nombre del servicio y versión de la API](#). Para obtener información sobre el AWS SDK for PHP versión 3 en GitHub, consulte [Recursos adicionales](#) (p. 317).

Enlaces externos: [Documento de API](#) | [GitHub](#) | [Gitter](#) | [Blog](#) | [Forum](#) | [Packagist](#)

Note

Si va a migrar el código de un proyecto de la versión 2 del SDK a la versión 3, lea primero [Actualización desde la versión 2 del AWS SDK para PHP](#) (p. 14).

Introducción

- [Requisitos y recomendaciones para el AWS SDK para PHP versión 3](#) (p. 4)
- [Instalación del AWS SDK para PHP versión 3](#) (p. 5)
- [Patrones de uso básicos de AWS SDK para PHP versión 3](#) (p. 7)
- [Actualización desde la versión 2 de AWS SDK para PHP](#) (p. 14)

Guías de SDK

- [Configuración de AWS SDK para PHP \(p. 24\)](#)
- [Credenciales para el AWS SDK para PHP \(p. 44\)](#)
- [Objetos de comando en el AWS SDK para PHP \(p. 61\)](#)
- [Promesas en el AWS SDK para PHP \(p. 66\)](#)
- [Controladores y middleware en AWS SDK para PHP \(p. 71\)](#)
- [Flujos en el AWS SDK para PHP \(p. 79\)](#)
- [Paginadores en el AWS SDK para PHP \(p. 82\)](#)
- [Esperadores en el AWS SDK para PHP \(p. 84\)](#)
- [JMESPath Expresiones en AWS SDK para PHP \(p. 86\)](#)
- [SDK Metrics \(p. 90\)](#)

Características específicas de servicios

- [Uso de AWS Cloud9 con el AWS SDK para PHP \(p. 100\)](#)
- [Uso del administrador de sesiones de DynamoDB con AWS SDK para PHP \(p. 102\)](#)
- [Cliente de Amazon S3 para varias regiones \(p. 107\)](#)
- [Encapsulador de flujo de Amazon S3 \(p. 108\)](#)
- [Gestor de transferencias de Amazon S3 \(p. 113\)](#)
- [Cifrado del lado del cliente de Amazon S3 \(p. 116\)](#)

Examples

- [Amazon CloudFront \(p. 122\)](#)
- [Firma de solicitudes personalizadas del dominio de AmazonCloudSearch \(p. 142\)](#)
- [Ejemplos de AmazonCloudWatch \(p. 144\)](#)
- [Amazon EC2 \(p. 161\)](#)
- [Firma de una solicitud de búsqueda de Amazon Elasticsearch Service \(p. 171\)](#)
- [Ejemplos de AWS Identity and Access Management \(p. 172\)](#)
- [AWS Key Management Service \(p. 190\)](#)
- [Ejemplos de Amazon Kinesis \(p. 207\)](#)
- [Ejemplos de AWS ElementalMediaConvert \(p. 219\)](#)
- [Ejemplos de Amazon S3 \(p. 227\)](#)
- [Ejemplos de Amazon Simple Email Service \(p. 254\)](#)
- [Ejemplos de Amazon SNS \(p. 278\)](#)
- [Ejemplos de Amazon SQS \(p. 292\)](#)

Para ver más ejemplos, consulte el [Catálogo de muestras de código de AWS](#).

Reference

- [PREGUNTAS FRECUENTES \(p. 309\)](#)

- [Glosario \(p. 315\)](#)
- [Contribución al SDK](#)
- [Documentación de Guzzle](#)

Envíe los problemas en GitHub:

- [Enviar problemas de documentación](#)
- [Informar de un error o solicitar una característica](#)

Documentación de la API

Busque la documentación de la API para el SDK en <http://docs.aws.amazon.com/aws-sdk-php/v3/api/>.

Mantenimiento y compatibilidad con las versiones principales del SDK

Para obtener más información sobre el mantenimiento y la compatibilidad con las versiones principales del SDK y sus dependencias subyacentes, consulte lo siguiente en la [Guía de referencia de credenciales y configuración compartida de AWS SDKs y las herramientas](#):

- [Política de mantenimiento de AWS SDKs y de herramientas](#)
- [Matriz de compatibilidad de la versión de AWS SDKs y de las herramientas](#)

Introducción al AWS SDK for PHP versión 3

Este capítulo se ha pensado para prepararle para utilizar AWS SDK for PHP versión 3.

Temas

- [Requisitos y recomendaciones para el AWS SDK for PHP versión 3 \(p. 4\)](#)
- [Instalación del AWS SDK for PHP versión 3 \(p. 5\)](#)
- [Patrones de uso básicos del AWS SDK for PHP versión 3 \(p. 7\)](#)
- [Actualización desde la versión 2 de la AWS SDK for PHP \(p. 14\)](#)

Requisitos y recomendaciones para el AWS SDK for PHP versión 3

Para obtener los mejores resultados con AWS SDK for PHP, asegúrese de que su entorno es compatible con los siguientes requisitos y recomendaciones.

Requirements

Para utilizar la AWS SDK for PHP, debe utilizar la versión 5.5.0 o posterior de PHP con la [extensión PHP de SimpleXML habilitada](#). Para firmar Amazon CloudWatch URLs privada, también necesita la [extensión PHP de OpenSSL](#).

Recommendations

Además de los requisitos mínimos, le recomendamos que también instale, desinstale y utilice lo siguiente.

Instale cURL 7.16.2 o posterior	Utilice una versión reciente de cURL compilada con OpenSSL/NSS y zlib. Si cURL no está instalado en su sistema y no configura un http_handler personalizado para su cliente, el SDK utilizará el encapsulador de flujo de PHP.
Use OPCache	Use la extensión OPcache para mejorar el rendimiento de PHP al almacenar código de bytes de script compilado previamente en la memoria compartida. Esto elimina la necesidad de PHP de cargar y analizar scripts en cada solicitud. Esta extensión suele estar habilitada de forma predeterminada. Con Amazon Linux, debe instalar el paquete yum php56-opcache o php55-opcache para utilizar la extensión OPcache.
Desinstale Xdebug	Xdebug le ayuda a identificar los cuellos de botella de rendimiento. Sin embargo, si el rendimiento es fundamental para la aplicación, no instale la extensión Xdebug en el entorno de producción.

	Cargar la extensión ralentiza el rendimiento del SDK de forma considerable.
Utilice un cargador automático de classmap Composer	Los cargadores automáticos cargan clases cuando un script PHP lo requiere. Composer genera un cargador automático que puede cargar automáticamente scripts de PHP de su aplicación y todos los demás scripts de PHP que precisa su aplicación, incluido AWS SDK for PHP. Para los entornos de producción, se recomienda utilizar un cargador automático de classmap para mejorar el rendimiento del cargador automático. Puede generar un cargador automático de classmap transfiriendo la opción <code>-o</code> o <code>--optimize-autoloader</code> al comando de instalación de Composer.

Prueba de compatibilidad

Ejecute el archivo `compatibility-test.php` en el SDK para comprobar que su sistema puede ejecutar el SDK. Además de cumplir los requisitos mínimos de sistema del SDK, la prueba de compatibilidad comprueba la configuración opcional y realiza recomendaciones que pueden ayudar a mejorar el rendimiento. La prueba de compatibilidad presenta los resultados en la línea de comandos o en un navegador web. Al revisar los resultados de la prueba en un navegador, las comprobaciones correctas aparecen en verde, las advertencias en morado y los errores en rojo. Si se muestra en la línea de comandos, el resultado de la comprobación aparece en una línea independiente.

Al informar sobre un problema con el SDK, compartir el resultado de la prueba de compatibilidad ayuda a identificar la causa subyacente.

Instalación del AWS SDK for PHP versión 3

Puede instalar AWS SDK for PHP versión 3:

- Como una dependencia mediante Composer
- Como phar empaquetado previamente del SDK
- Como un archivo ZIP del SDK

Antes de instalar AWS SDK for PHP versión 3, asegúrese de que su entorno esté utilizando PHP versión 5.5 o posterior. Obtenga más información sobre los [requisitos y las recomendaciones del entorno](#) (p. 4).

Note

La instalación del SDK a través de los métodos `.phar` y `.zip` requiere que la [extensión PHP de cadena multibyte](#) se instale y habilite por separado.

Instalar AWS SDK for PHP como una dependencia mediante Composer

[Composer](#) es la forma recomendada de instalar AWS SDK for PHP. Composer es una herramienta para PHP que administra e instala las dependencias del proyecto.

Para obtener más información acerca de cómo instalar Composer, configure la carga automática y siga otras prácticas recomendadas para definir dependencias. Para ello, consulte getcomposer.org.

Instalación de Composer

Si Composer aún no está en su proyecto, descargue e [instale Composer](#).

Para Windows, descargue y ejecute el archivo [Composer-Setup.exe](#).

Para Linux, siga la instalación de la línea de comandos en [la página de descarga de Composer](#).

Añadir AWS SDK for PHP como una dependencia mediante Composer

Si [Composer ya está instalado globalmente](#) en su sistema, ejecute lo siguiente en el directorio base del proyecto para instalar AWS SDK for PHP como una dependencia:

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

De lo contrario, escriba este comando de Composer para instalar la última versión de AWS SDK for PHP como una dependencia.

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

Añadir un cargador automático a sus scripts de PHP

Para utilizar AWS SDK for PHP en sus scripts, incluya el cargador automático en los scripts, tal y como se indica a continuación.

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

Instalación con el phar empaquetado

Cada versión de AWS SDK for PHP incluye un archivo phar (PHP) empaquetado previamente que contiene todas las clases y dependencias que necesita para ejecutar el SDK. Además, el phar registra automáticamente un cargador automático de clases para AWS SDK for PHP y todas sus dependencias.

Puede [descargar el phar empaquetado](#) e incluirlo en sus scripts.

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

Note

No se recomienda utilizar archivos PHP con el parche Suhosin, aunque es común en distribuciones de Ubuntu y de Debian. En este caso, es posible que tenga que habilitar el uso de archivos phar en el archivo suhosin.ini. Si no lo hace e incluye un archivo phar en el código, se generará un error silencioso. Para modificar suhosin.ini, añada la siguiente línea.

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

Instalación con el archivo ZIP

AWS SDK for PHP incluye un archivo ZIP que contiene todas las clases y dependencias que necesita para ejecutar el SDK. Además, el archivo ZIP incluye un cargador automático de clases para AWS SDK for PHP y sus dependencias.

Para instalar el SDK, [descargue el archivo .zip](#) y, a continuación, extráigalo en su proyecto en la ubicación que elija. A continuación, incluya el cargador automático en sus scripts, tal y como se indica a continuación.

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

Patrones de uso básicos del AWS SDK for PHP versión 3

Este tema se centra en los patrones de uso básico de AWS SDK for PHP.

Prerequisites

- [Descargar e instalar el SDK. \(p. 5\)](#)
- Recuperar sus [claves de acceso de AWS](#).

Inclusión del SDK en su código

Independientemente de la técnica utilizada para instalar el SDK, puede incluir el SDK en su código con tan solo una única instrucción `require`. Consulte la siguiente tabla para saber qué código PHP se adapta mejor a su técnica de instalación. Sustituya las instancias de `/path/to/` por la ruta real de su sistema.

Técnica de instalación	Instrucción require
Uso de Composer	<code>require '/path/to/vendor/autoload.php';</code>
Uso de phar	<code>require '/path/to/aws.phar';</code>
Uso de ZIP	<code>require '/path/to/aws-autoloader.php';</code>

En este tema mostramos ejemplos que presuponen que se ha utilizado el método de instalación de Composer. Si utiliza un método de instalación diferente, puede consultar esta sección para encontrar el código `require` correcto que debe usar.

Resumen de uso

Para utilizar el SDK para interactuar con un servicio de AWS, cree una instancia de un objeto Client. Los objetos de cliente tienen métodos que se corresponden directamente con operaciones de la API del servicio. Para ejecutar una operación determinada, llame al método correspondiente. Este método devuelve un objeto de resultado similar a una matriz cuando se ejecuta correctamente o genera una excepción en caso de error.

Crear un cliente

Puede crear un cliente pasando una matriz asociativa de opciones a un constructor de clientes.

Importaciones

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

use Aws\S3\S3Client;

use Aws\Exception\AwsException;
```

Código de muestra

```
//Create a S3Client
$s3 = new Aws\S3\S3Client([
    'version' => 'latest',
    'region' => 'us-east-2'
]);
```

Tenga en cuenta que no hemos proporcionado credenciales de forma explícita al cliente. Esto se debe a que el SDK debería detectar las credenciales a partir de las [variables de entorno \(p. 46\)](#) (mediante `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY`), un [archivo INI de credenciales de AWS \(p. 46\)](#) en su directorio HOME, las [credenciales del perfil de instancia de AWS Identity and Access Management \(IAM\) \(p. 48\)](#) o los [proveedores de credenciales \(p. 52\)](#).

Todas las opciones de configuración generales del cliente se describen en detalle en la [guía de configuración \(p. 24\)](#). La matriz de opciones proporcionada a un cliente puede variar en función del cliente que esté creando. Estas opciones de configuración de cliente personalizadas se describen en la [documentación de la API](#) de cada cliente.

Usar la clase Sdk

La clase `Aws\Sdk` actúa como una fábrica de clientes y se utiliza para administrar las opciones de configuración compartidas entre varios clientes. Las mismas opciones que se pueden proporcionar a un constructor de clientes específico también se pueden proporcionar a la clase `Aws\Sdk`. Estas opciones se aplican después a cada constructor de clientes.

Importaciones

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

use Aws\S3\S3Client;

use Aws\Exception\AwsException;
```

Código de muestra

```
// 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();
```

Las opciones que se comparten entre todos los clientes se colocan en pares clave-valor en el nivel de raíz. Los datos de configuración específicos del servicio se pueden proporcionar en una clave que es igual al espacio de nombres de un servicio (por ejemplo, "S3", "DynamoDb", etc.).

```
$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();
```

Los valores de configuración específicos del servicio son una combinación de los valores específicos del servicio y los valores del nivel de raíz (es decir, los valores específicos del servicio se combinan suavemente con los valores del nivel de raíz).

Note

Le recomendamos encarecidamente que utilice la clase `Sdk` para crear clientes si utiliza varias instancias de cliente en su aplicación. La clase `Sdk` utiliza automáticamente el mismo cliente HTTP para cada cliente del SDK, lo que permite que los clientes del SDK de diferentes servicios puedan realizar solicitudes HTTP sin que haya bloqueo. Si los clientes del SDK no utilizan el mismo cliente HTTP, es posible que las solicitudes HTTP enviadas por el cliente del SDK bloqueen la orquestación de promesas entre los servicios.

Ejecutar operaciones de servicio

Puede ejecutar una operación de servicio utilizando el método del mismo nombre en un objeto de cliente. Por ejemplo, para realizar la [operación PutObject de Amazon S3](#), debe llamar al método `Aws\S3\S3Client::putObject()`.

Importaciones

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

use Aws\S3\S3Client;
```

Código de muestra

```
// 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'];
```

Las operaciones disponibles para un cliente y la estructura de la entrada y la salida se definen en el tiempo de ejecución en función de un archivo de descripción del servicio. Al crear un cliente, debe proporcionar una versión (por ejemplo, "2006-03-01" o "latest" (la más reciente)). El SDK buscará el archivo de configuración correspondiente en función de la versión proporcionada.

Todos los métodos de operación del tipo `putObject()` aceptan un único argumento: una matriz asociativa que representa los parámetros de la operación. La estructura de esta matriz (y la estructura del objeto de resultado) se define para cada operación en la documentación de la API del SDK (por ejemplo, consulte la documentación de la API para la operación [putObject](#)).

Opciones del controlador HTTP

También puede ajustar la forma en que el controlador HTTP subyacente ejecuta la solicitud mediante el uso del parámetro `@http` especial. Las opciones que puede incluir en el parámetro `@http` son las mismas que puede establecer al crear instancias para el cliente con la [opción de cliente "http"](#) (p. 32).

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

Solicitudes asíncronas

Puede enviar comandos de forma simultánea utilizando las características asíncronas del SDK. Puede enviar solicitudes de forma asíncrona agregando el sufijo `Async` al nombre de una operación. Esto inicia la solicitud y devuelve una promesa. La promesa se cumple cuando el objeto de resultado es correcto o se rechaza con una excepción en caso de error. Esto le permite crear varias promesas y hacer que envíen solicitudes HTTP de forma simultánea cuando el controlador HTTP subyacente transfiere las solicitudes.

Importaciones

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

Código de muestra

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

```
$sdk = new Aws\Sdk([
    'region' => 'us-west-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();
```

Puede forzar una promesa para que se complete de forma síncrona utilizando el método `wait` de la promesa. Forzar que se complete la promesa también "desencapsula" el estado de la promesa de forma predeterminada, lo que significa que, o bien devolverá el resultado de la promesa, o lanzará la excepción que se detectó. Al llamar a `wait()` en una promesa, el proceso se bloquea hasta que se completa la solicitud HTTP y el resultado se rellena o se lanza una excepción.

Si utiliza el SDK con una biblioteca de bucles de evento, no bloquee los resultados. En su lugar, utilice el método `then()` de un resultado para obtener acceso a una promesa que se resuelve o rechaza cuando se completa la operación.

Importaciones

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

Código de muestra

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region' => 'us-west-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();
    });
}
```

Uso de objetos Result

Si se ejecuta una operación correctamente, se devuelve un objeto `Aws\Result`. En lugar de devolver los datos XML o JSON sin procesar de un servicio, el SDK fuerza que los datos de la respuesta aparezcan en una estructura de matriz asociativa. El SDK normaliza algunos aspectos de los datos en función de su conocimiento del servicio en concreto y la estructura de respuesta subyacente.

Puede obtener acceso a los datos del objeto `AWSResult` como una matriz PHP asociativa.

Importaciones

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

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

Código de muestra

```
// 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();
```

El contenido del objeto de resultado depende de la operación que se ha ejecutado y de la versión del servicio. La estructura del resultado de cada operación API se documenta en la documentación de la API de cada operación.

El SDK se integra con [JMESPath](#), un [DSL](#) que se utiliza para buscar y manipular datos JSON o, en nuestro caso, matrices PHP. El objeto de resultado contiene un método `search()` que puede utilizar para extraer datos del resultado de forma más declarativa.

Código de muestra

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

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

Gestión de errores

Gestionar errores síncronos

Si se produce un error al realizar una operación, se genera una excepción. Por este motivo, si necesita administrar errores en el código, utilice bloques `try/catch` en torno a sus operaciones. El SDK lanza excepciones específicas del servicio cuando se produce un error.

El siguiente ejemplo utiliza la `Aws\S3\S3Client`. Si se produce un error, la excepción lanzada será del tipo `Aws\S3\Exception\S3Exception`. Todas las excepciones específicas de servicios que lanza el SDK son una extensión de la clase `Aws\Exception\AwsException`. Esta clase contiene información útil sobre el error, incluido el ID de la solicitud, el código de error y el tipo de error. Tenga en cuenta que para algunos servicios compatibles, los datos de respuesta se convierten en una estructura matriz asociativa (similar a los objetos de `Aws\Result`) que se puede evaluar como una matriz asociativa de PHP normal. El método `toArray()` devolverá cualquier dato, en caso de que exista.

Importaciones

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

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

Código de muestra

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region' => 'us-west-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";

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

Gestionar errores asíncronos

Al enviar solicitudes asíncronas no se lanzan excepciones. En lugar de ello, debe utilizar el método `then()` u `otherwise()` de la promesa devuelta para recibir el resultado o el error.

Importaciones

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

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

Código de muestra

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

Puede "desencapsular" la promesa y hacer que se lance la excepción en su lugar.

Importaciones

Importaciones

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

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

Código de muestra

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

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

Actualización desde la versión 2 de laAWS SDK for PHP

En este tema se muestra cómo migrar su código para utilizar la versión 3 de AWS SDK for PHP y las diferencias de la nueva versión frente a la versión 2 del SDK.

Note

El patrón de uso básico del SDK (es decir, `$result = $client->operation($params);`) no ha cambiado de la versión 2 a la 3, por lo que la migración debería realizarse sin problema.

Introduction

La versión 3 de AWS SDK for PHP representa una mejora importante de las capacidades del SDK, incorpora los comentarios realizados por nuestros clientes a lo largo de dos años, la actualización de nuestras dependencias, la mejora del rendimiento y la adopción de los estándares más recientes de PHP.

¿Qué novedades incluye la versión 3?

A partir de este momento, la versión 3 de AWS SDK for PHP sigue los [estándares PSR-4 y PSR-7](#) y el estándar [SemVer](#).

Otras nuevas características

- Sistema de middleware para personalizar el comportamiento del cliente del servicio
- Paginadores flexibles para iterar a través de los resultados paginados
- Capacidad para consultar datos de objetos de resultado y paginador con JMESPath
- Depuración sencilla mediante la opción de configuración 'debug'

Capa HTTP desacoplada

- [Guzzle 6](#) se utiliza de forma predeterminada para enviar solicitudes, pero también se admite Guzzle 5.
- El SDK funcionará en entornos donde cURL no está disponible.
- También se admiten los controladores HTTP personalizados.

Solicitudes asíncronas

- Las características como los esperadores y los cargadores multiparte también se pueden utilizar de forma asíncrona.
- Los flujos de trabajo asíncronos se pueden crear utilizando promesas y corutinas.
- El rendimiento de las solicitudes simultáneas o en lotes ha mejorado.

¿Cuáles son las diferencias con la versión 2?

Las dependencias del proyecto están actualizadas

En esta versión, las dependencias del SDK han cambiado.

- El SDK ahora requiere PHP 5.5 y superior. Usamos numerosos [generadores](#) dentro del código del SDK.
- Hemos actualizado el SDK para utilizar [Guzzle 6](#) (o 5), que proporciona la implementación del cliente HTTP subyacente que utiliza el SDK para enviar solicitudes a los servicios de AWS. La versión más reciente de Guzzle incluye una serie de mejoras, que engloba solicitudes asíncronas, controladores HTTP intercambiables, conformidad con PSR-7, mejora del rendimiento y mucho más.
- El paquete PSR-7 de PHP-FIG ([psr/http-message](#)) define interfaces para que representen solicitudes HTTP, respuestas HTTP, URLs y flujos. Estas interfaces se utilizan en el SDK y en Guzzle, y permiten operar internamente con otros paquetes compatibles con PSR-7.
- La implementación de PSR-7 ([guzzlehttp/psr7](#)) de Guzzle proporciona una implementación de las interfaces en PSR-7 y varias clases y funciones útiles. Tanto el SDK como Guzzle 6 dependen fuertemente de este paquete.
- La implementación de [Promises/A+](#) de Guzzle ([guzzlehttp/promises](#)) se utiliza tanto en el SDK como en Guzzle para proporcionar interfaces para gestionar solicitudes y corutinas asíncronas. Mientras el controlador HTTP multi-cURL de Guzzle implementa en última instancia el modelo E/S sin bloqueo que permite las solicitudes asíncronas, este paquete ofrece la posibilidad de programar dentro de ese paradigma. Para obtener información detallada, consulte [Promesas en el AWS SDK para PHP versión 3 \(p. 66\)](#).
- La implementación PHP de [JMESPath](#) ([mt Dowling/jmespath.php](#)) se utiliza en el SDK para proporcionar la capacidad de consulta de datos de los métodos `Aws\Result::search()` y `Aws\ResultPaginator::search()`. Encontrará expresiones [JMESPath en el AWS SDK para PHP versión 3 \(p. 86\)](#) para obtener más información.

Ahora se requieren las opciones Region y Version

Al crear una instancia de un cliente para cualquier servicio, especifique las opciones `'region'` y `'version'`. En la versión 2 de AWS SDK for PHP, `'version'` era totalmente opcional y `'region'`, a veces. En la versión 3, ambas son siempre necesarias. Al ser explícito sobre ambas opciones, le permite bloquear la versión de la API y la región de AWS sobre las que está codificando. Cuando se creen versiones de la API nuevas o haya nuevas regiones de AWS disponibles, estará aislado de posibles cambios bruscos hasta que esté listo para actualizar su configuración de forma explícita.

Note

Si no le preocupa la versión de la API que está utilizando, establezca la opción `'version'` en `'latest'`. Sin embargo, le recomendamos que establezca los números de versión de la API de forma explícita para el código de producción.

No todos los servicios están disponibles en todas las regiones de AWS. Para ver una lista de las regiones disponibles, consulte la referencia [Regiones y puntos de enlace](#).

En el caso de los servicios que solo están disponibles a través de un único punto de enlace global (por ejemplo, Amazon Route 53, AWS Identity and Access Management y Amazon CloudFront), cree instancias de los clientes con su región establecida en `us-east-1`.

Important

El SDK también incluye clientes en varias regiones, que pueden enviar solicitudes a diferentes regiones de AWS en función de un parámetro (`@region`) suministrado como parámetro del comando. El parámetro `Region` que utilizan estos clientes de forma predeterminada se especifica en la opción `region` suministrada al constructor de clientes.

La creación de instancias del cliente utiliza el constructor

En la versión 3 de AWS SDK for PHP, la forma en la que crea la instancia de un cliente ha cambiado. En lugar de utilizar los métodos `factory` de la versión 2, aquí puede crear una instancia de un cliente utilizando la palabra clave `new`.

```
use Aws\DynamoDb\DynamoDbClient;

// Version 2 style
$client = DynamoDbClient::factory([
    'region' => 'us-east-2'
]);

// Version 3 style
$client = new DynamoDbClient([
    'region' => 'us-east-2',
    'version' => '2012-08-10'
]);
```

Note

De todos modos, también se puede crear una instancia de un cliente utilizando el método `factory()`. Sin embargo, se considera obsoleto.

La configuración del cliente ha cambiado

Las opciones de configuración del cliente en la versión 3 de AWS SDK for PHP han cambiado un poco respecto a la versión 2. Consulte [Configuración del AWS SDK para PHP versión 3 \(p. 24\)](#) para ver una descripción de todas las opciones admitidas.

Important

En la versión 3, las opciones `'key'` y `'secret'` ya no son válidas en el nivel de raíz, pero las puede transferir como parte de la opción `'credentials'`. Uno de los motivos por lo que lo hemos hecho es para evitar que los desarrolladores codifiquen de forma rígida sus credenciales de AWS en sus proyectos.

El objeto Sdk

La versión 3 de AWS SDK for PHP presenta el objeto `Aws\Sdk` en sustitución de `Aws\Common\Aws`. El objeto `Sdk` actúa como una fábrica de clientes y se utiliza para administrar las opciones de configuración compartidas en varios clientes.

Aunque la clase `Aws` de la versión 2 del SDK funcionaba como un localizador de servicio (siempre devolvía la misma instancia de un cliente), la clase `Sdk` de la versión 3 devuelve una nueva instancia de un cliente cada vez que se utiliza.

El objeto `Sdk` tampoco admite el mismo formato de archivo de configuración de la versión 2 del SDK. Dicho formato de configuración era específico de Guzzle 3 y ya está obsoleto. La configuración se puede simplificar con matrices y se documenta en [Uso de la clase Sdk \(p. 8\)](#).

Algunos resultados de la API han cambiado

Para ser coherentes en la forma en que el SDK analiza el resultado de una operación de la API, ahora Amazon ElastiCache, Amazon RDS y Amazon Redshift tienen un elemento de encapsulamiento adicional en algunas respuestas de la API.

Por ejemplo, ahora al llamar al Amazon RDS `DescribeEngineDefaultParameters` se obtiene la versión 3 de , se incluye un elemento de encapsulamiento "EngineDefaults". En la versión 2, este elemento no existía.

```
$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'];
```

Estas son las operaciones afectadas y ahora contienen un elemento de encapsulamiento en la salida del resultado (incluido a continuación entre paréntesis):

- Amazon ElastiCache
 - `AuthorizeCacheSecurityGroupIngress` (`CacheSecurityGroup`)
 - `CopySnapshot` (instantánea)
 - `CreateCacheCluster` (`CacheCluster`)
 - `CreateCacheParameterGroup` (`CacheParameterGroup`)
 - `CreateCacheSecurityGroup` (`CacheSecurityGroup`) de
 - `CreateCacheSubnetGroup` (`CacheSubnetGroup`)
 - `CreateReplicationGroup` (`ReplicationGroup`) de
 - `CreateSnapshot` (instantánea)
 - `DeleteCacheCluster` (`CacheCluster`)
 - `DeleteReplicationGroup` (`ReplicationGroup`) de
 - `DeleteSnapshot` (instantánea)
 - `DescribeEngineDefaultParameters` (`EngineDefaults`)
 - `ModifyCacheCluster` (`CacheCluster`)
 - `ModifyCacheSubnetGroup` (`CacheSubnetGroup`)
 - `ModifyReplicationGroup` (`ReplicationGroup`) de
 - `PurchaseReservedCacheNodesOffering` (`ReservedCacheNode`)
 - `RebootCacheCluster` (`CacheCluster`)
 - `RevokeCacheSecurityGroupIngress` (`CacheSecurityGroup`)

- Amazon RDS
 - AddSourceIdentifierToSubscription (EventSubscription)
 - AuthorizeDBSecurityGroupIngress (DBSecurityGroup)
 - CopyDBParameterGroup (DBParameterGroup)
 - CopyDBSnapshot (instantánea de DBS)
 - CopyOptionGroup (OptionGroup)
 - CreateDBInstance (DBInstance)
 - CreateDBInstanceReadReplica (DBInstance)
 - CreateDBParameterGroup (DBParameterGroup)
 - CreateDBSecurityGroup (DBSecurityGroup)
 - CreateDBSnapshot (DBSnapshot)
 - CreateDBSubnetGroup (DBSubnetGroup) de
 - CreateEventSubscription (EventSubscription)
 - CreateOptionGroup (OptionGroup)
 - DeleteDBInstance (DBInstance)
 - DeleteDBSnapshot (instantánea de DBS)
 - DeleteEventSubscription (EventSubscription)
 - DescribeEngineDefaultParameters (EngineDefaults)
 - ModifyDBInstance (DBInstance)
 - ModifyDBSubnetGroup (DBSubnetGroup) de
 - ModifyEventSubscription (EventSubscription)
 - ModifyOptionGroup (OptionGroup)
 - PromoteReadReplica (DBInstance)
 - PurchaseReservedDBInstancesOffering (ReservedDBInstance)
 - RebootDBInstance (DBInstance)
 - RemoveSourceIdentifierFromSubscription (EventSubscription)
 - RestoreDBInstanceFromDBSnapshot (DBInstance)
 - RestoreDBInstanceToPointInTime (DBInstance)
 - RevokeDBSecurityGroupIngress (DBSecurityGroup)
 - Amazon Redshift
 - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
 - AuthorizeSnapshotAccess (instantánea)
 - CopyClusterSnapshot (instantánea)
 - CreateCluster (clúster)
 - CreateClusterParameterGroup (ClusterParameterGroup)
 - CreateClusterSecurityGroup (ClusterSecurityGroup)
 - CreateClusterSnapshot (instantánea)
 - CreateClusterSubnetGroup (ClusterSubnetGroup)
 - CreateEventSubscription (EventSubscription)
 - CreateHsmClientCertificate (HsmClientCertificate)
 - CreateHsmConfiguration (HsmConfiguration)
 - DeleteCluster (clúster)
 - DeleteClusterSnapshot (instantánea)
-
- DescribeDefaultClusterParameters (DefaultClusterParameters)
 - DisableSnapshotCopy (clúster)
 - EnableSnapshotCopy (clúster)

- `ModifyCluster` (clúster)
- `ModifyClusterSubnetGroup` (`ClusterSubnetGroup`)
- `ModifyEventSubscription` (`EventSubscription`)
- `ModifySnapshotCopyRetentionPeriod` (clúster)
- `PurchaseReservedNodeOffering` (`ReservedNode`)
- `RebootCluster` (clúster)
- `RestoreFromClusterSnapshot` (clúster)
- `RevokeClusterSecurityGroupIngress` (`ClusterSecurityGroup`)
- `RevokeSnapshotAccess` (instantánea)
- `RotateEncryptionKey` (clúster)

Se han eliminado las clases Enum

Hemos eliminado las clases `Enum` (por ejemplo, `Aws\S3\Enum\CannedAcl`) de la versión 2 de AWS SDK for PHP. Las clases `Enum` eran clases concretas dentro de la API pública del SDK que incluían constantes que representan grupos de valores de parámetros válidos. Dado que estos objetos `Enum` son específicos de las versiones de la API, pueden cambiar con el paso del tiempo, pueden entrar en conflicto con palabras reservadas de PHP y acabar no siendo muy útiles, los hemos eliminado de la versión 3. Esto es compatible con la naturaleza agnóstica de la versión de la API y basada en datos de la versión 3.

En lugar de utilizar valores de objetos `Enum`, utilice los valores literales directamente (por ejemplo, `CannedAcl::PUBLIC_READ` → `'public-read'`).

Se han eliminado las clases de excepción detalladas

Hemos eliminado las clases de excepción detalladas que existían en los espacios de nombres de cada servicio (por ejemplo, `Aws\Rds\Exception\{SpecificError}Exception`) por motivos muy similares por los que hemos eliminado los objetos `Enum`. Las excepciones que lanza un servicio u operación dependen de la versión de la API que se utiliza (pueden cambiar de versión a versión). Además, la lista completa de excepciones que puede lanzar una operación determinada no está disponible, lo que provocaba que las clases de excepción detalladas de la versión 2 fueran incompletas.

Gestione los errores capturando la clase de excepción raíz de cada servicio (por ejemplo, `Aws\Rds\Exception\RdsException`). Puede utilizar el método `getAwsErrorCode()` de la excepción para comprobar si hay códigos de error específicos. Desde el punto de vista funcional, es similar a la captura de diferentes clases de excepción pero proporciona dicha función sin añadir sobredimensionamiento al SDK.

Se han eliminado las clases Facade estáticas

En la versión 2 de AWS SDK for PHP, había una característica oculta inspirada por Laravel que permitía llamar a `enableFacades()` en la clase `Aws` para habilitar el acceso estático a los distintos clientes de servicios. Esta característica es contraria a las prácticas recomendadas de PHP, por lo que dejamos de documentarla hace más de un año. En la versión 3, esta característica se ha eliminado por completo. Recupere sus objetos de cliente del objeto `Aws\Sdk` y utilícelos como instancias de objeto, no como clases estáticas.

Los paginadores sustituyen a los iteradores

La versión 2 de AWS SDK for PHP tenía una característica llamada **iteradores**. Estos objetos se utilizaban para la iteración de los resultados paginados. Una de las quejas que recibimos al respecto era que no eran suficientemente flexibles, ya que el iterador solo emitía valores específicos de cada resultado. Si necesitaba otros valores de los resultados, solo se podían recuperar a través de los agentes de escucha.

En la versión 3, los iteradores se han sustituido por [Paginadores \(p. 82\)](#). Su finalidad es similar, pero los paginadores son más flexibles. Esto se debe a que generaban objetos Result en lugar de valores de una respuesta.

En los siguientes ejemplos se muestra la diferencia entre los paginadores y los iteradores, y se explica cómo recuperar resultados paginados para la operación S3 `ListObjects` tanto en la versión 2 como en la versión 3.

```
// 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";
    }
}
```

Los objetos del paginador tienen un método `search()` que le permite utilizar expresiones [JMESPath \(p. 86\)](#) para extraer datos más fácilmente del conjunto de resultados.

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'my-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

Note

El método `getIterator()` sigue siendo compatible para que la transición a la versión 3 sea suave, pero le recomendamos que migre su código para utilizar paginadores.

Han cambiado muchas abstracciones de nivel superior

En general, muchas de las abstracciones de nivel superior (objetos auxiliares específicos de servicios, aparte de los clientes) se han mejorado o actualizado. Y algunas se han eliminado.

- Actualizaciones:
 - La forma en que utiliza la [Carga multiparte de Amazon S3 \(p. 233\)](#) ha cambiado. Amazon S3 Glacier La carga multiparte de se ha modificado de formas similares.
 - La forma de crear [Amazon S3 URLs prefirado \(p. 241\)](#) ha cambiado.
 - El espacio de nombres `Aws\S3\Sync` se ha sustituido por la clase `Aws\S3\Transfer`. Los métodos `S3Client::uploadDirectory()` y `S3Client::downloadBucket()` siguen estando disponibles, pero tienen diferentes opciones. Consulte la documentación del [Gestor de transferencias de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 113\)](#).
 - `Aws\S3\Model\ClearBucket` y `Aws\S3\Model\DeleteObjectsBatch` se han sustituido por `Aws\S3\BatchDelete` y `S3Client::deleteMatchingObjects()`.
 - El uso de las opciones y los comportamientos del [uso del administrador de sesiones de DynamoDB con el AWS SDK para PHP versión 3 \(p. 102\)](#) han cambiado ligeramente.
 - El espacio de nombres `Aws\DynamoDb\Model\BatchRequest` se ha sustituido por `Aws\DynamoDb\WriteRequestBatch`. Obtenga la documentación de [DynamoDB WriteRequestBatch](#).

- `Aws\Ses\SesClient` ahora se encarga de la codificación base64 de `RawMessage` cuando se utiliza la operación `SendRawEmail`.
- Eliminaciones:
 - Las clases `Item`, `Attribute` y `ItemIterator` de Amazon DynamoDB ya eran obsoletas en la [versión 2.7.0](#).
 - El validador de mensajes de Amazon SNS es ahora [un proyecto ligero independiente](#) que no requiere el SDK como dependencia. Sin embargo, este proyecto se incluye en las distribuciones Phar y ZIP del SDK. Encontrará una guía de introducción [en el blog de desarrollo de PHP de AWS](#).
 - Se han eliminado `AcpBuilder` de Amazon S3 y los objetos relacionados.

Comparación de ejemplos de código de ambas versiones del SDK

Los siguientes ejemplos muestran algunos ejemplos de cómo trabajar con la versión 3 de AWS SDK for PHP puede variar respecto a cómo se trabaja con la versión 2.

Ejemplo: Amazon S3 Operación `ListObjects`

En la versión 2 del SDK

```
<?php

require '/path/to/vendor/autoload.php';

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

$s3 = S3Client::factory([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'my-bucket-name',
        'Key'    => 'my-object-key'
    ]);

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

En la versión 3 del SDK

Diferencias clave:

- Se utiliza `new` en lugar de `factory()` para crear instancias del cliente.
- Las opciones `'version'` y `'region'` son obligatorias para crear instancias.

```
<?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";
}
```

Ejemplo: Creación de una instancia de un cliente con configuración global

En la versión 2 del SDK

```
<?php return array(
    'includes' => array('_aws'),
    'services' => array(
        'default_settings' => array(
            'params' => array(
                'profile' => 'my_profile',
                'region'  => 'us-east-1'
            )
        ),
        'dynamodb' => array(
            'extends' => 'dynamodb',
            'params' => array(
                'region' => 'us-west-2'
            )
        ),
    )
);
```

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\Common\Aws;

$aws = Aws::factory('path/to/my/config.php');

$sqs = $aws->get('sqs');
// Note: SQS client will be configured for us-east-1.

$dynamodb = $aws->get('dynamodb');
// Note: DynamoDB client will be configured for us-west-2.
```

En la versión 3 del SDK

Diferencias clave:

- Se utiliza la clase `Aws\Sdk` en lugar de `Aws\Common\Aws`.
- No hay ningún archivo de configuración. En su lugar, utilice una matriz para configurar.
- La opción `'version'` es obligatoria durante la creación de instancias.
- Utilice los métodos `create<Service>()` en lugar de `get('<service>')`.

```
<?php

require '/path/to/vendor/autoload.php';

$sdk = new Aws\Sdk([
    'profile' => 'my_profile',
    'region' => 'us-east-1',
    'version' => 'latest',
    'DynamoDb' => [
        'region' => 'us-west-2',
    ],
]);

$sqs = $sdk->createSqs();
// Note: Amazon SQS client will be configured for us-east-1.

$dynamodb = $sdk->createDynamoDb();
// Note: DynamoDB client will be configured for us-west-2.
```

Configuración del AWS SDK for PHP versión 3

AWS SDK for PHP se compone de varias características y componentes. Cada uno de los siguientes temas describe los componentes que se utilizan en el SDK.

Temas

- [Configuración del AWS SDK for PHP versión 3 \(p. 24\)](#)
- [Credenciales para el AWS SDK for PHP versión 3 \(p. 44\)](#)
- [Objetos comando en el AWS SDK for PHP versión 3 \(p. 61\)](#)
- [Promesas en el AWS SDK for PHP versión 3 \(p. 66\)](#)
- [Controladores y middleware en el AWS SDK for PHP versión 3 \(p. 71\)](#)
- [Flujos en el AWS SDK for PHP versión 3 \(p. 79\)](#)
- [Paginadores en el AWS SDK for PHP versión 3 \(p. 82\)](#)
- [Esperadores en el AWS SDK for PHP versión 3 \(p. 84\)](#)
- [Expresiones de JMESPath en el AWS SDK for PHP versión 3 \(p. 86\)](#)
- [Métricas de SDK en la AWS SDK for PHP versión 3 \(p. 90\)](#)

Configuración del AWS SDK for PHP versión 3

En esta guía se describen las opciones del constructor de clientes. Estas opciones se pueden proporcionar en un constructor de clientes o suministrar a la clase `Aws\Sdk`. La matriz de opciones proporcionada a un tipo de cliente específico puede variar en función del cliente que esté creando. Estas opciones de configuración de cliente personalizadas se describen en la [documentación de la API](#) de cada cliente.

Tenga en cuenta que algunas opciones de configuración comprobarán y utilizarán valores predeterminados basados en variables de entorno o en un archivo de configuración de AWS. De forma predeterminada, el archivo de configuración que se está comprobando será `.aws/config` en su directorio principal, normalmente `~/ .aws/config`. Sin embargo, puede utilizar la variable de entorno `AWS_CONFIG_FILE` para establecer dónde se encuentra la ubicación predeterminada del archivo de configuración. Esto puede ser especialmente útil si está restringiendo el acceso a archivos a ciertos directorios con `open_basedir` y similares.

Opciones de configuración

- [api_provider \(p. 25\)](#)
- [credentials \(p. 25\)](#)
- [debug \(p. 27\)](#)
- [stats \(p. 28\)](#)
- [endpoint \(p. 29\)](#)
- [endpoint_provider \(p. 29\)](#)
- [endpoint_discovery \(p. 30\)](#)
- [handler \(p. 31\)](#)

- [http](#) (p. 32)
- [http_handler](#) (p. 37)
- [profile](#) (p. 38)
- [region](#) (p. 39)
- [retries](#) (p. 39)
- [scheme](#) (p. 41)
- [service](#) (p. 41)
- [signature_provider](#) (p. 42)
- [signature_version](#) (p. 42)
- [ua_append](#) (p. 42)
- [use_aws_shared_config_files](#) ([usar_archivos_de_configuración_compartidos_de_aws](#)) (p. 43)
- [validate](#) (p. 43)
- [version](#) (p. 44)

En el siguiente ejemplo se muestra cómo transferir opciones a un constructor de clientes de Amazon S3.

```
use Aws\S3\S3Client;

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

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

Consulte la [guía de uso básico](#) (p. 7) para obtener más información sobre cómo crear clientes.

api_provider

Tipo

callable

Es una función invocable de PHP que acepta un argumento de tipo, servicio y versión, y que devuelve una matriz de datos de configuración correspondiente. El valor del tipo puede ser: `api`, `waiter` o `paginator`.

De forma predeterminada, el SDK usa una instancia de `Aws\Api\FileSystemApiProvider` que carga archivos API desde la carpeta `src/data` del SDK.

credentials

Tipo

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

Transfiera un objeto `Aws\Credentials\CredentialsInterface` para utilizar una instancia de credenciales específicas.

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



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

Si no proporciona una opción `credentials`, el SDK intentará cargar las credenciales desde su entorno en el orden que se indica a continuación:

1. Cargar las credenciales de las [variables de entorno \(p. 46\)](#).
2. Cargar las credenciales desde un [archivo.ini de credenciales \(p. 46\)](#).
3. Cargar las credenciales desde un [rol de IAM \(p. 48\)](#).

Transfiera `false` para utilizar credenciales nulas y no firmar solicitudes.

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

Transfiera una función invocable de [proveedor de credenciales \(p. 52\)](#) para crear las credenciales utilizando una función.

```
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
]);
```

Transfiera las credenciales almacenadas en caché a una instancia de `Aws\CacheInterface` para almacenar en caché los valores devueltos por la cadena de proveedor predeterminada en varios procesos.

```
use Aws\Credentials\CredentialProvider;
use Aws\DoctrineCacheAdapter;
use Doctrine\Common\Cache\ApcuCache;

$cache = new DoctrineCacheAdapter(new ApcuCache);
$provider = CredentialProvider::defaultProvider();
$cachedProvider = CredentialProvider::cache($provider, $cache);

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

Puede encontrar más información acerca de cómo proporcionar credenciales a un cliente en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#).

Note

Las credenciales se cargan y validan lentamente cuando están en uso.

debug

Tipo

`bool|array`

Emite información de depuración acerca de cada transferencia. La información de depuración contiene información sobre cada cambio de estado de una transacción, ya que se prepara y se envía a través de la red. En el resultado de la depuración también se incluye información sobre el controlador HTTP específico que utiliza un cliente (por ejemplo, depurar el resultado de cURL).

Establezca este parámetro en `true` para mostrar información de depuración al enviar solicitudes.

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

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

De forma alternativa, puede proporcionar una matriz asociativa con las siguientes claves.

logfn (invocable)

Es una función que se invoca con los mensajes de registro. De forma predeterminada, se utiliza la función `echo` de PHP.

stream_size (entero)

Cuando el tamaño de un flujo es superior a este número, los datos del flujo no se registran. Establezca el valor en 0 para no registrar datos de flujo.

scrub_auth (bool)

Establezca este parámetro en `false` para deshabilitar la limpieza de los datos de automatización de los mensajes registrados (es decir, que su ID de clave de acceso de AWS y la firma se transferirán a través de `logfn`).

http (bool)

Establezca este parámetro en `false` para deshabilitar la función de "depuración" de los controladores HTTP de nivel inferior (por ejemplo, el resultado de cURL detallado).

auth_headers (array)

Establezca este parámetro en un mapeo de clave-valor de encabezados que desea sustituir mapeados al valor por el que los desea sustituir. Estos valores no se utilizan a menos que `scrub_auth` se haya establecido en `true`.

auth_strings (array)

Establezca este parámetro en un mapeo de clave-valor de expresiones regulares para mapearlos a sus reemplazos. El cepillo de datos de autenticación utiliza estos valores si `scrub_auth` se establece en `true`.

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

La salida de depuración es extremadamente útil para diagnosticar problemas en AWS SDK for PHP. Proporcione la salida de depuración para un caso de error aislado al abrir problemas en el SDK.

stats

Tipo

bool|array

Vincula estadísticas de transferencia a errores y resultados devueltos por las operaciones del SDK.

Establezca este parámetro en `true` para recopilar las estadísticas de transferencia de las solicitudes enviadas.

```
$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'];
```

De forma alternativa, puede proporcionar una matriz asociativa con las siguientes claves.

retries (bool)

Establezca este parámetro en `true` para habilitar los informes sobre los reintentos. Las estadísticas de reintento se recopilan de forma predeterminada y se devuelven.

http (bool)

Establezca este parámetro en `true` para habilitar la recopilación de estadísticas de adaptadores HTTP de nivel inferior (por ejemplo, valores devueltos en `GuzzleHttpTransferStats`). Los controladores HTTP deben admitir la opción `__on_transfer_stats` para que surta efecto. Las estadísticas HTTP se devuelven como una matriz indexada de matrices asociativas; cada matriz asociativa contiene las estadísticas de transferencia devueltas para una solicitud por el controlador HTTP del cliente. Está deshabilitado de forma predeterminada.

Si se ha reintentado una solicitud, se devolverán las estadísticas de transferencia de cada solicitud. `$result['@metadata']['transferStats']['http'][0]` incluirá las estadísticas de la primera

solicitud, `$result['@metadata']['transferStats']['http'][1]`, las estadísticas de la segunda solicitud y así sucesivamente.

timer (bool)

Establezca este parámetro en `true` para habilitar un temporizador de comandos que informa sobre el tiempo de reloj total que ha necesitado una operación para ejecutarse en segundos. Está deshabilitado de forma predeterminada.

```
$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

Tipo

string

Es el URI completo del servicio web. Esto es necesario para los servicios, como [MediaConvert](#), que utilizan puntos de enlace específicos de la cuenta. Para estos servicios, solicite este punto de enlace mediante el método `:doc`describeEndpoints`<emc-examples-getendpoint>`.

Esto solo es necesario cuando se conecta a un punto de enlace personalizado (por ejemplo, una versión local de Amazon S3 o [Amazon DynamoDB Local](#)).

A continuación se muestra un ejemplo de la conexión a Amazon DynamoDB Local:

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

Para ver una lista de las regiones y puntos de enlace de AWS disponibles, consulte [Regiones y puntos de enlace de AWS](#).

endpoint_provider

Tipo

callable

Es una función invocable de PHP opcional que acepta un hash de opciones, incluidas una clave "service" y "region". Devuelve NULL o un hash de datos de punto de enlace, de los cuales se requiere la clave "endpoint".

A continuación se muestra un ejemplo de cómo crear un proveedor de punto de enlace mínimo.

```
$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

Tipo

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

La detección de puntos de enlace identifica y se conecta al punto de enlace correcto para una API de servicio que admita la detección de puntos de enlace. En el caso de los servicios que admiten, pero no requieren la detección de puntos de enlace, habilite `endpoint_discovery` durante la creación del cliente. Si un servicio no admite la detección de puntos de enlace, esta configuración no se tiene en cuenta.

Aws\EndpointDiscovery\ConfigurationInterface

Es un proveedor de configuración opcional que permite la conexión automática al punto de enlace adecuado de una API de servicio para las operaciones que especifique el servicio.

El objeto `Aws\EndpointDiscovery\Configuration` acepta dos opciones: un valor booleano, "enabled", que indica si la detección de puntos de enlace está habilitada y un número entero, "cache_limit", que indica el número máximo de claves en la caché de puntos de enlace.

Para cada cliente creado, pase un objeto `Aws\EndpointDiscovery\Configuration` para utilizar una configuración específica para la detección de puntos de enlace.

```
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,  
]);
```

Pase una instancia de `Aws\CacheInterface` para almacenar en caché los valores devueltos por la detección de puntos de enlace en varios procesos.

```
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),
]);
```

Pase una matriz a la detección de puntos de enlace.

```
use Aws\S3\S3Client;

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

handler

Tipo

callable

Es un controlador que acepta un objeto de comando y objeto de solicitud, y que devuelve una promesa (GuzzleHttp\Promise\PromiseInterface) que se cumple con un objeto Aws\ResultInterface o se rechaza con una Aws\Exception\AwsException. Un controlador no acepta un siguiente controlador, ya que es terminal y se espera que lleve a cabo un comando. Si no se proporciona ningún controlador, se utiliza un controlador Guzzle predeterminado.

Puede utilizar el parámetro Aws\MockHandler para devolver los resultados simulados o lanzar excepciones simuladas. El usuario debe poner los resultados o las excepciones a la cola, y MockHandler los saca de la cola siguiendo el orden FIFO.

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

$mock = new MockHandler();

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

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

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-west-2',
```

```
'version' => 'latest',  
'handler' => $mock  
]);  
  
// Result object response will contain ['foo' => 'bar']  
$result = $client->listTables();  
  
// This will throw the exception that was enqueued  
$client->listTables();
```

http

Tipo

array

Establezca este parámetro en una matriz de opciones HTTP que se aplica a las solicitudes y transferencias HTTP creadas por el SDK.

El SDK es compatible con las siguientes opciones de configuración:

cert

Tipo

string|array

Especifique el certificado de cliente con formato PEM.

- Establecer como una cadena de texto para la ruta de acceso solo al archivo del certificado.

```
use Aws\S3\S3Client;  
  
$client = new S3Client([  
    'region' => 'us-west-2',  
    'version' => 'latest',  
    'http' => ['cert' => '/path/to/cert.pem']  
]);
```

- Establecer como una matriz que contiene la ruta y la contraseña.

```
use Aws\S3\S3Client;  
  
$client = new S3Client([  
    'region' => 'us-west-2',  
    'version' => 'latest',  
    'http' => [  
        'cert' => ['/path/to/cert.pem', 'password']  
    ]  
]);
```

connect_timeout

Es un valor flotante que describe el número de segundos que se debe esperar al intentar conectarse a un servidor. Use 0 para esperar de forma indefinida (es el comportamiento por defecto).

```
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
    ]
]);
```

debug

Tipo

`bool|resource`

Indica al controlador HTTP subyacente que emita información de depuración. La información de depuración que proporcionan los diferentes controladores HTTP variará.

- Transfiera `true` para escribir la salida de depuración en STDOUT.
- Transfiera un `resource` tal como lo devuelve `fopen` para escribir la salida de depuración en un recurso de flujo de PHP específico.

decode_content

Tipo

`bool`

Indica al controlador HTTP subyacente que infle el cuerpo de respuestas comprimidas. Si no está habilitado, los cuerpos de respuestas comprimidos pueden inflarse con un `GuzzleHttp\Psr7\InflateStream`.

Note

La decodificación de contenido está habilitada de forma predeterminada en el controlador HTTP predeterminado del SDK. Por motivos de compatibilidad con versiones anteriores, este valor predeterminado no se puede cambiar. Si almacena archivos comprimidos en Amazon S3, le recomendamos que deshabilite contenido decodificando en el nivel de cliente de S3.

```
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

Tipo

`int`

Es el número de milisegundos de retraso antes de enviar la solicitud. Se utiliza a menudo para retrasar antes de volver a realizar una solicitud.

expect

Tipo

`bool|string`

Esta opción se pasa directamente al controlador HTTP subyacente. De forma predeterminada, espere: El encabezado 100-continue se establece cuando el cuerpo de la solicitud supera 1 MB. `true` o `false` habilita o deshabilita el encabezado en todas las solicitudes. Si se utiliza un número entero, utilizarán el encabezado únicamente las solicitudes cuyo cuerpo supere este valor. Cuando se utiliza un número entero, si el tamaño del cuerpo es desconocido, se enviará el encabezado Expect.

Warning

Si se deshabilita el encabezado Expect, es posible que el servicio no sea capaz de devolver errores de autenticación o de otro tipo. Esta opción debe configurarse con precaución.

progress

Tipo

`callable`

Define la función a invocar cuando se realiza el progreso de la transferencia. La función acepta los argumentos siguientes:

1. El número total de bytes previsto que se va a descargar.
2. El número de bytes descargado hasta el momento.
3. El número de bytes previsto que se va a cargar.
4. El número de bytes cargado hasta el momento.

```
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

Tipo

string|array

Puede conectarse a un servicio de AWS a través de un proxy utilizando la opción `proxy`.

- Proporcione un valor de cadena para conectarse a un proxy para todos los tipos de URIs. El valor de cadena de proxy puede contener un esquema, nombre de usuario y contraseña. Por ejemplo, "http://username:password@192.168.16.1:10".
- Proporcione una matriz de configuración de proxy asociativa donde la clave sea el esquema del URI y el valor sea el proxy del URI en cuestión (es decir, que puede introducir diferentes proxies para los puntos de enlace "http" y "https").

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

Puede utilizar la variable de entorno `HTTP_PROXY` para configurar un proxy específico del protocolo "http" y la variable de entorno `HTTPS_PROXY` para configurar un proxy específico de "https".

sink

Tipo

resource|string|Psr\Http\Message\StreamInterface

La opción `sink` controla el lugar donde se descargan los datos de respuesta de una operación.

- Proporcione un `resource` tal como lo devuelve `fopen` para descargar el cuerpo de la respuesta en un flujo de PHP.
- Proporcione la ruta a un archivo en el disco como valor `string` para descargar el cuerpo de la respuesta en un archivo específico en el disco.
- Proporcione una `Psr\Http\Message\StreamInterface` para descargar el cuerpo de la respuesta en un objeto de flujo de PSR determinado.

Note

El SDK descarga el cuerpo de la respuesta en un flujo de PHP temporal de forma predeterminada. Esto significa que los datos se guardan en la memoria hasta que el tamaño del cuerpo alcanza 2 MB, momento en el que los datos se escriben en un archivo temporal en el disco.

synchronous

Tipo

`bool`

La opción `synchronous` informa al controlador HTTP subyacente que el usuario está intentando bloquear el resultado.

stream

Tipo

`bool`

Establezca este parámetro en `true` para indicar al controlador HTTP subyacente que desea transmitir el cuerpo de la respuesta del servicio web, en lugar de descargarlo directamente. Por ejemplo, esta opción se utiliza en la clase del encapsulador de flujo de Amazon S3 para garantizar que se transmiten los datos.

timeout

Tipo

`float`

Es un valor flotante que describe el tiempo de espera de la solicitud en segundos. Use 0 para esperar de forma indefinida (es el comportamiento por defecto).

```
use Aws\DynamoDb\DynamoDbClient;

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

verify

Tipo

`bool|string`

Puede personalizar el comportamiento de verificación del certificado SSL/TLS homólogo del SDK utilizando la opción `verify http`.

- Establezca este parámetro en `true` para habilitar la verificación del certificado homólogo SSL/TLS y utilizar el paquete de CA predeterminado proporcionado por el sistema operativo.
- Establezca este parámetro en `false` para deshabilitar la verificación del certificado homólogo. (¡No es seguro!)
- Establezca este parámetro en una cadena para proporcionar la ruta a un paquete de certificados de CA para habilitar la verificación utilizando un paquete de CA personalizado.

Si no se encuentra el paquete de CA para su sistema y recibe un error, deberá proporcionar la ruta a un paquete de CA al SDK. Si no necesita un paquete de CA específico, Mozilla proporciona un paquete de CA de uso común que puede descargar [aquí](#) (lo gestionará el gestor de cURL). En cuanto tenga un paquete de CA disponible en el disco, puede establecer el valor del `.ini` de PHP `openssl.cafile` de modo que apunte hacia la ruta al archivo, lo que le permitirá omitir la opción de solicitud `verify`. Encontrará más detalles sobre certificados SSL en el [sitio web de cURL](#).

```
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

Tipo

`callable`

La opción `http_handler` se utiliza para integrar el SDK con otros clientes HTTP. Una opción `http_handler` es una función que acepta un objeto `Psr\Http\Message\RequestInterface` y una matriz de opciones `http` que se aplica al comando, y que devuelve un objeto `GuzzleHttp\Promise\PromiseInterface` que se ha completado con un objeto `Psr\Http\Message\ResponseInterface` o rechazado con una matriz con los siguientes datos de excepción:

- `exception`: (`\Exception`) es la excepción que se ha encontrado.

- `response`: (`Psr\Http\Message\ResponseInterface`) es la respuesta que se ha recibido (si es que se ha recibido alguna).
- `connection_error`: (`bool`) se establece en `true` para marcar el error como error de conexión. Si se establece este valor en `true` también permite al SDK que vuelva a intentar la operación de forma automática, si es necesario.

El SDK convierte automáticamente el `http_handler` dado en una opción `handler` normal encapsulando el `http_handler` proporcionado con un objeto `Aws\WrappedHttpHandler`.

De forma predeterminada, el SDK utiliza Guzzle como su controlador HTTP. Puede proporcionar un controlador HTTP diferente aquí, o proporcionar un cliente Guzzle con sus propias opciones definidas personalizadas.

Configuración de la versión de TLS

Un caso de uso es establecer la versión de TLS utilizada por Guzzle con Curl, suponiendo que Curl esté instalado en su entorno. Tenga en cuenta las [las restricciones de la versión](#) de Curl para qué versión de TLS es compatible. Por defecto, se utiliza la última versión. Si la versión TLS está establecida explícitamente y el servidor remoto no admite esta versión, se producirá un error en lugar de usar una versión de TLS anterior.

Puede determinar la versión de TLS que se está utilizando para una operación de cliente determinada estableciendo la opción de cliente `debug` en `true` y examinando el resultado de la conexión SSL. Esa línea podría ser similar a esto: `SSL connection using TLSv1.2`

Ejemplo de configuración de TLS 1.2 con Guzzle 6:

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

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

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

Note

La opción `handler` sustituye a cualquier opción `http_handler` proporcionada.

profile

Tipo

`string`

Le permite especificar qué perfil utilizar cuando se crean las credenciales del archivo de credenciales de AWS en el directorio HOME. Este valor sustituye a la variable de entorno `AWS_PROFILE`.

Note

Si se especifica "profile" se omitirá la clave "credentials".

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

Para obtener más información sobre la configuración de credenciales y el formato del archivo .ini, consulte [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#).

region

Tipo

string

Obligatorio

true

Es la región de AWS a la que hay que conectarse. Para obtener una lista de las regiones disponibles, consulte [Regiones y puntos de enlace de AWS](#).

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

retries

Tipo

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

Predeterminado

int(3)

Configura el modo de reintento y el número máximo de reintentos permitidos para un cliente. Transfiera o para deshabilitar los reintentos.

Los tres modos de reintento son: * *legacy*: la implementación de reintentos heredada predeterminada * *standard* añade un sistema de cuotas de reintentos para evitar reintentos que son improbables

que tengan éxito

- *adaptive*: se basa en el modo estándar, añadiendo un limitador de velocidad del lado del cliente. Tenga en cuenta que este modo se considera experimental.

La configuración para reintentos consiste en el modo y los intentos máximos que se van a utilizar para cada solicitud. La configuración se puede establecer en un par de ubicaciones diferentes, en el siguiente orden de prioridad.

Orden de prioridad

El orden de prioridad para la configuración de reintento es el siguiente (1 anula 2-3, etc.):

1. Opción de configuración de cliente
2. Variables de entorno
3. Archivo de configuración compartida de AWS

Variables de entorno

- `AWS_RETRY_MODE`: establecido en `legacy`, `standard` o `adaptive`.
- `AWS_MAX_ATTEMPTS`: establecido en un valor entero para el máximo de intentos por solicitud

Claves de archivos de configuración compartida

- `retry_mode`: establecido en `legacy`, `standard` o `adaptive`.
- `max_attempts`: establecido en un valor entero para el máximo de intentos por solicitud

Configuración de cliente

El siguiente ejemplo deshabilita los reintentos para el cliente de Amazon DynamoDB.

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

El siguiente ejemplo transfiere un número entero, que pasará de forma predeterminada al modo `legacy` con el número de reintentos transferidos

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

El objeto **`Aws\Retry\Configuration`** acepta dos parámetros, el modo de reintento

y un número entero para el máximo de intentos por solicitud. En este ejemplo se transfiere un

objeto `Aws\Retry\Configuration` para la configuración de reintento.

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

$enabled = true;
$cache_limit = 1000;

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

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

```
'retries' => $config,
]);
```

En este ejemplo se transfiere una matriz para la configuración de reintento.

```
use Aws\S3\S3Client;

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

En este ejemplo se transfiere una instancia de `Aws\CacheInterface` para almacenar en caché los valores devueltos por el proveedor de configuración de reintento predeterminado.

```
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),
]);
```

scheme

Tipo

string

Predeterminado

string(5) "https"

Esquema de URI que se debe utilizar para establecer la conexión. El SDK utiliza puntos de enlace "https" (es decir, que utiliza conexiones SSL/TLS) de forma predeterminada. Puede intentar conectarse a un servicio a través de un punto de enlace "http" sin cifrar configurando `scheme` en "http".

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

Consulte [Regiones y puntos de enlace de AWS](#) para obtener una lista de los puntos de enlace y saber si un servicio es compatible con el esquema http.

service

Tipo

string

Obligatorio

true

Es el nombre del servicio que se debe utilizar. Este valor se suministra de forma predeterminada cuando se utiliza un cliente proporcionado por el SDK (es decir, `Aws\S3\S3Client`). Esta opción es útil al probar un servicio que todavía no se ha publicado en el SDK pero que está disponible en el disco.

signature_provider

Tipo

callable

Es una función invocable que acepta un nombre de versión de firma (por ejemplo, `v4`), un nombre de servicio y una región de AWS, y que devuelve un objeto `Aws\Signature\SignatureInterface` o `NULL` si el proveedor es capaz de crear un signatario para los parámetros especificados. Este proveedor se utiliza para crear los signatarios que utiliza el cliente.

El SDK proporciona distintas funciones en la clase `Aws\Signature\SignatureProvider` que se pueden utilizar para crear proveedores de firma personalizada.

signature_version

Tipo

string

Es una cadena que representa una versión de firma personalizada para utilizarla con un servicio (por ejemplo, `v4`, etc.). La versión de firma por operación PUEDE sustituir a esta versión de firma solicitada, si es necesario.

Los siguientes ejemplos muestran cómo configurar un cliente de Amazon S3 para utilizar la [versión de firma 4](#):

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

Note

El `signature_provider` que utiliza su cliente DEBE poder crear la opción `signature_version` que proporcione. El `signature_provider` predeterminado que utiliza el SDK puede crear objetos de firma para las versiones de firma `"v4"` y `"anonymous"`.

ua_append

Tipo

string|string[]

Predeterminado

[]

Es una cadena o matriz de cadenas que se añade a la cadena de agente-usuario que se transfiere al controlador HTTP.

use_aws_shared_config_files (usar_archivos_de_configuración_compartidos_de_aws)

Tipo

bool|array

Predeterminado

bool(true)

Establézcalo en false para deshabilitar la comprobación del archivo de configuración compartido en "~/aws/config" y "~/aws/credentials". Esto anulará la variable de entorno AWS_CONFIG_FILE.

validate

Tipo

bool|array

Predeterminado

bool(true)

Establezca este parámetro como false para deshabilitar la validación de parámetros del lado del cliente. Es posible que si desactiva la validación mejore ligeramente el rendimiento del cliente, pero la diferencia es insignificante.

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

Establezca este parámetro como matriz asociativa de opciones de validación para habilitar restricciones de validación específicas:

- **required:** compruebe que los parámetros obligatorios están presentes (activos de forma predeterminada).
- **min:** valide la longitud mínima de un valor (activada de forma predeterminada).
- **max:** valide la longitud máxima de un valor.
- **pattern:** valide que el valor coincide con una expresión regular.

```
// Validate only that required values are present
$s3 = new Aws\S3\S3Client([
    'version' => '2006-03-01',
```

```
'region'    => 'eu-west-1',  
'validate' => ['required' => true]  
]);
```

version

Tipo

string

Obligatorio

true

Es la versión del servicio web a utilizar (por ejemplo, 2006-03-01).

Se precisa un valor de configuración "version". Especificar una restricción de versión garantiza que el código no se vea afectado por un cambio importante en el servicio. Por ejemplo, si utiliza Amazon S3, puede bloquear su versión de la API en 2006-03-01.

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

Puede encontrar una lista de las versiones de la API disponibles en la [página de la documentación de la API](#) de cada cliente. Si no consigue cargar una versión de la API específica, es posible que tenga que actualizar su copia del SDK.

Puede proporcionar la cadena `latest` con el valor de configuración "version" para que utilice la versión de la API más reciente disponible que encuentre el proveedor de API de su cliente (el `api_provider` predeterminado escanea el directorio `src/data` del SDK para buscar modelos de API).

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

Warning

No le recomendamos utilizar `latest` en una aplicación de producción, ya que integrar una nueva versión secundaria del SDK que incluya una actualización de la API podría romper su aplicación de producción.

Credenciales para el AWS SDK for PHP versión 3

Para realizar solicitudes a Amazon Web Services, proporcione [claves de acceso de AWS](#), también conocidas como credenciales, al AWS SDK for PHP.

Puede hacerlo de las siguientes maneras:

- Utilice la cadena predeterminada de proveedores de credenciales (recomendado).
- Utilice un proveedor de credenciales específico o una cadena de proveedores (o cree el suyo propio).
- Proporcione usted mismo las credenciales. Pueden ser las credenciales de la cuenta raíz, las credenciales de IAM o credenciales temporales obtenidas de AWS STS.

Important

Por motivos de seguridad, es absolutamente recomendable que utilice usuarios de IAM en lugar de la cuenta raíz para el acceso a AWS. Para obtener más información, consulte [Prácticas recomendadas de IAM](#) en la IAM User Guide.

Uso de la cadena predeterminada de proveedores de credenciales

Cuando se inicializa un cliente de servicio sin proporcionar argumentos de credenciales, el SDK utiliza la cadena predeterminada de proveedores de credenciales para encontrar las credenciales de AWS. El SDK utiliza el primer proveedor de la cadena que devuelve unas credenciales sin errores.

La cadena predeterminada de proveedores busca y utiliza las credenciales en el orden indicado a continuación:

1. [Utilizar las credenciales de las variables de entorno \(p. 46\)](#).

La configuración de las variables de entorno resulta útil si está realizando trabajos de desarrollo en un equipo que no sea una instancia Amazon EC2.

2. [Utilizar los perfiles y el archivo de credenciales compartidas de AWS \(p. 46\)](#).

Este archivo de credenciales es el mismo que utilizan otros SDKs y la AWS CLI. Si ya está usando un archivo de credenciales compartidas, puede utilizarlo para este fin.

Este método se utiliza en la mayoría de nuestros ejemplos de código PHP.

3. [Asumir un rol de IAM \(p. 48\)](#).

Los roles de IAM proporcionan a las aplicaciones de la instancia credenciales de seguridad temporales para realizar llamadas de AWS. Por ejemplo, los roles de IAM ofrecen una forma sencilla de distribuir y administrar credenciales en varias instancias Amazon EC2.

Otras formas de añadir credenciales

También puede añadir credenciales como se indica a continuación:

- [Utilizando un proveedor de credenciales \(p. 52\)](#).

Proporcione lógica personalizada para las credenciales cuando cree el cliente.

- [Utilizando credenciales temporales desde AWS STS \(p. 59\)](#).

Cuando utilice un token de Multi-Factor Authentication (MFA) para la autenticación de dos factores, utilice AWS STS para conceder al usuario credenciales temporales y para obtener acceso a los servicios de AWS o utilizar el AWS SDK for PHP.

- [Utilizando credenciales codificadas de forma rígida \(p. 60\)](#) (no se recomienda).

Warning

Codificar de forma rígida sus credenciales puede ser peligroso, ya que es fácil confirmar accidentalmente las credenciales en un repositorio de SCM. Esto puede exponer potencialmente sus credenciales a más personas de las que usted pretendía. También puede dificultar la rotación de credenciales en el futuro. No envíe código con credenciales codificadas de forma rígida al control de código fuente.

- [Creando clientes anónimos \(p. 61\)](#).

Cree un cliente que no esté asociado a ningunas credenciales cuando el servicio permita el acceso anónimo.

Para obtener más información, consulte las prácticas recomendadas de [Credenciales de seguridad de AWS](#) en la [Referencia general de Amazon Web Services](#).

Uso de credenciales desde las variables de entorno

Si utiliza variables de entorno para guardar las credenciales, evitará compartir accidentalmente su clave de acceso secreta de AWS. Recomendamos que nunca añada sus claves de acceso de AWS directamente al cliente en ningún archivo de producción. Las cuentas de muchos desarrolladores se han visto comprometidas por la filtración de sus claves.

Para autenticarse en Amazon Web Services, el SDK primero busca las credenciales en las variables de entorno. El SDK utiliza la función `getenv()` para buscar las variables de entorno `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `AWS_SESSION_TOKEN`. Estas credenciales se denominan credenciales de entorno.

Si aloja la aplicación en [AWS Elastic Beanstalk](#), puede configurar las variables de entorno `AWS_ACCESS_KEY_ID` y `AWS_SECRET_KEY` a través de la consola de AWS Elastic Beanstalk, de forma que el SDK pueda utilizar dichas credenciales de forma automática.

También puede definir las variables de entorno en la línea de comandos, tal y como se muestra aquí.

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
# The access key for your AWS account.
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFcYEXAMPLEKEY
# 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/bPxrFcYEXAMPLEKEY
# 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.
```

Uso del archivo de credenciales de AWS y de los perfiles de credenciales

Un archivo de credenciales es un archivo de texto no cifrado que contiene las claves de acceso. El archivo debe:

- Estar en la misma máquina en la que se ejecuta la aplicación.
- Denominarse `credentials`.
- Encontrarse en la carpeta `.aws/` del directorio principal.

El directorio principal puede variar según el sistema operativo. En Windows, puede hacer referencia al directorio principal mediante la variable de entorno `%UserProfile%`. Con los sistemas tipo Unix, puede utilizar la variable de entorno `$HOME` o `~` (tilde).

Si ya utiliza este archivo para otras herramientas de SDKs y AWS CLI (como la), no es necesario cambiar nada para utilizar los archivos en este SDK. Si utiliza credenciales distintas para cada herramienta o aplicación, puede utilizar perfiles para configurar varias claves de acceso en el mismo archivo de configuración.

Este método se utiliza en todos nuestros ejemplos de código PHP.

El uso de un archivo de credenciales de AWS ofrece los siguientes beneficios:

- Las credenciales de los proyectos se almacenan fuera de estos, por lo que es imposible confirmarlas por error en el control de versiones.
- Puede definir y asignar un nombre a varios conjuntos de credenciales en un solo lugar.
- Es posible reutilizar con facilidad las mismas credenciales en distintos proyectos.
- es compatible con otras herramientas y SDKs de AWS, este mismo archivo de credenciales. Esto le permite volver a utilizar sus credenciales con otras herramientas.

El formato del archivo de credenciales de AWS debería parecerse al formato siguiente.

```
[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
```

Cada sección (por ejemplo, `[default]`, `[project1]`), representa un perfil de credenciales independiente. Puede hacer referencia a los perfiles desde un archivo de configuración del SDK o al crear una instancia de un cliente, mediante la opción `profile`.

```
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'
]);
```

Si no se han proporcionado explícitamente credenciales o perfiles al SDK y no se han definido credenciales en las variables de entorno, pero sí se ha definido un archivo de credenciales, el SDK utiliza el perfil "predeterminado". Puede cambiar el perfil predeterminado especificando un nombre de perfil alternativo en la variable de entorno `AWS_PROFILE`.

Asumir rol con perfil

Puede configurar AWS SDK para PHP para que utilice un rol de IAM mediante la definición un perfil para el rol en `~/.aws/credentials`.

Cree un nuevo perfil con `role_arn` para el rol que asumirá. Incluya también el `source_profile` de un perfil con credenciales que tenga permisos para asumir el rol de IAM.

Perfil en `~/.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
```

Al establecer la variable de entorno `AWS_PROFILE`, o la opción `profile` al crear la instancia de un cliente, se asumirá el rol especificado en `project1`, utilizando el perfil `default` como credenciales de origen.

También se pueden asumir roles para los perfiles definidos en `~/.aws/config`. El establecimiento de la variable de entorno `AWS_SDK_LOAD_NONDEFAULT_CONFIG` permite cargar perfiles para asumir un rol desde `~/.aws/config`. Cuando se habilita, se cargarán los perfiles de `~/.aws/config` y `~/.aws/credentials`. Los perfiles de `~/.aws/credentials` se cargan en último lugar y tienen prioridad sobre un perfil de `~/.aws/config` con el mismo nombre. Los perfiles de cualquier ubicación pueden servir como `source_profile` o el perfil que se asumirá.

Perfil en `~/.aws/config`:

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

Perfil en `~/.aws/credentials`:

```
[project2]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
```

Con los archivos anteriores, se asumirá `[project1]` mediante `[project2]` como las credenciales de origen.

Cómo asumir roles de IAM

Uso de roles de IAM para credenciales variables de instancias Amazon EC2

Si está ejecutando una aplicación en una instancia Amazon EC2, la mejor manera de proporcionar credenciales para realizar llamadas a AWS es utilizar un [rol de IAM](#) para obtener credenciales de seguridad temporales.

Si utiliza roles de IAM, no tiene que ocuparse de la administración de credenciales desde la aplicación. Permiten que una instancia "asuma" un rol recuperando credenciales temporales del servidor de metadatos de la instancia Amazon EC2.

Las credenciales temporales, que suelen conocerse como credenciales del perfil de instancia, permiten acceder a las acciones y a los recursos que permite la política del rol. Amazon EC2 administra todas las

operaciones de autenticación segura de las instancias en el servicio de IAM para asumir el rol, así como las operaciones de actualización periódica de las credenciales del rol recuperadas. Esto mantiene su aplicación segura prácticamente sin esfuerzo por su parte.

Note

Las credenciales del perfil de instancia y otras credenciales temporales que AWS Security Token Service (AWS STS) genera no son compatibles con todos los servicios. Para saber si el servicio que utiliza es compatible con el uso de credenciales temporales, consulte la sección sobre [servicios de AWS compatibles con AWS STS](#).

Para evitar llegar siempre al servicio de metadatos, puede pasar una instancia de `Aws\CacheInterface` como opción `'credentials'` a un constructor de clientes. De este modo, el SDK utiliza las credenciales del perfil de instancia en su lugar. Para obtener información detallada, consulte [Configuración de AWS SDK para PHP versión 3 \(p. 24\)](#).

Creación y asignación de un rol de IAM a una instancia Amazon EC2

1. Cree un cliente de IAM.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Iam\IamClient;
```

Código de muestra

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

2. Cree un rol de IAM con los permisos necesarios para las acciones y los recursos que vaya a utilizar.

Código de muestra

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

3. Cree un perfil de instancia de IAM y guarde el nombre de recurso de Amazon (ARN) del resultado.

Note

Si utiliza la consola de IAM en lugar del AWS SDK for PHP, la consola crea automáticamente un perfil de instancia y le asigna el mismo nombre que el rol con el que se corresponde.

Código de muestra

```
$IPN = 'InstanceProfileName';  
  
$result = $client->createInstanceProfile([  
    'InstanceProfileName' => $IPN ,  
]);  
  
$ARN = $result['Arn'];  
$InstanceID = $result['InstanceProfileId'];
```

4. Cree un cliente de Amazon EC2.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

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

5. Añada el perfil de instancia a una instancia Amazon EC2 en ejecución o detenida. Utilice el nombre del perfil de instancia del rol de IAM.

Código de muestra

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

Para obtener más información, consulte [Roles de IAM para Amazon EC2](#).

Uso de roles de IAM para tareas de Amazon ECS

Si utiliza roles de IAM para las tareas de Amazon Elastic Container Service (Amazon ECS), puede especificar un rol de IAM que pueden utilizar los contenedores de una tarea. Esta es una estrategia de administración de credenciales para que las utilicen las aplicaciones, similar al modo en que los perfiles de instancia Amazon EC2 proporcionan credenciales a las instancias Amazon EC2.

En lugar de crear y distribuir las credenciales de AWS a los contenedores o de utilizar el rol de la instancia Amazon EC2, puede asociar un rol de IAM con la definición de tareas de ECS o con la operación `RunTask` de la [API](#).

Note

Las credenciales del perfil de instancia y otras credenciales temporales generadas por AWS STS no son compatibles con todos los servicios. Para saber si el servicio que utiliza es compatible con el uso de credenciales temporales, consulte la sección sobre [servicios de AWS compatibles con AWS STS](#).

Para obtener más información, consulte los [funciones de IAM para tareas de Amazon EC2 Container Service](#).

Asumir un rol de IAM en otra cuenta de AWS

A la hora de trabajar en una cuenta de AWS (cuenta A) y desea asumir un rol en otra cuenta (cuenta B), primero debe crear un rol de IAM en la cuenta B. Este rol permite a las entidades de la cuenta (cuenta A) realizar acciones específicas en la cuenta B. Para obtener más información sobre el acceso entre cuentas, consulte [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles](#).

Después de crear un rol en Cuenta B, registre los ARN de rol. Usará este ARN cuando asuma el rol de Cuenta A. El rol se asume con las credenciales de AWS asociadas con su entidad en Cuenta A.

Cree un cliente de AWS STS con credenciales para su cuenta de AWS. A continuación, utilizamos un perfil de credenciales, pero puede utilizar cualquier método. Con el cliente de AWS STS recién creado, llame a `assume-role` y proporcione un `sessionName` personalizado. Recupere las nuevas credenciales temporales del resultado. De forma predeterminada, las credenciales duran una hora.

Código de muestra

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

Para obtener más información, consulte [Uso de roles de IAM](#) o [AssumeRole](#) en la Referencia de la API de AWS SDK for PHP.

Uso de un rol de IAM con la identidad web

La identidad web federada permite a los clientes utilizar proveedores de identidad de terceros para la autenticación al obtener acceso a los recursos de AWS. Para poder asumir un rol con identidad web, debe crear un rol de IAM y configurar un proveedor de identidad web (IdP). Para obtener más información, consulte [Creating a Role for Web Identity or OpenID Connect Federation \(Console\)](#).

Después de [crear un proveedor de identidad](#) y [crear un rol para su identidad web](#), utilice un cliente de AWS STS para autenticar a un usuario. Proporcione los valores de `webIdentityToken` y `ProviderId` para su identidad y el ARN del rol de IAM con permisos para el usuario.

Código de muestra

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

Para obtener más información, consulte : [federación a través de un proveedor de identidad](#) `webAssumeRoleWithWebIdentity` o `AssumeRoleWithWebIdentity` en la referencia de la API de AWS SDK for PHP.

Uso de un proveedor de credenciales

Un proveedor de credenciales es una función que devuelve un `GuzzleHttp\Promise\PromiseInterface` que se ejecuta con una instancia `Aws\Credentials\CredentialsInterface` o se rechaza con un `Aws\Exception\CredentialsException`. Puede utilizar los proveedores de credenciales para implementar su propia lógica personalizada para crear credenciales u optimizar la carga de credenciales.

Los proveedores de credenciales se transfieren a la opción de constructor de clientes `credentials`. Los proveedores de credenciales son asíncronos, lo que obliga a evaluarlos lentamente cada vez que se invoca una operación de la API. Por ello, si se pasa una función de proveedor de credenciales a un constructor de cliente del SDK, no se validan inmediatamente las credenciales. Si el proveedor de credenciales no devuelve un objeto de credenciales, se rechazará una operación de la API con una `Aws\Exception\CredentialsException`.

```
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
]);
```

Proveedores integrados en el SDK

El SDK proporciona varios proveedores integrados que se pueden combinar con los proveedores personalizados que se deseen.

Important

Los proveedores de credenciales se invocan cada vez que se ejecuta una operación de la API. Si la carga de credenciales es una tarea costosa (por ejemplo, la carga se realiza desde un

disco o un recurso de red) o si el proveedor no guarda en caché las credenciales, considere la posibilidad de encapsular el proveedor de credenciales en una función `Aws\Credentials\CredentialProvider::memoize`. El proveedor de credenciales predeterminado que utiliza el SDK se memoriza automáticamente.

Proveedor de `assumeRole`

Si utiliza `Aws\Credentials\AssumeRoleCredentialProvider` para crear credenciales asumiendo un rol, debe proporcionar la información de `'client'` con un objeto `StsClient` y con los detalles de `'assume_role_params'`, tal como se muestra a continuación.

Note

Para evitar tener que obtener de forma innecesaria credenciales de AWS STS siempre que ejecuta una operación de la API, puede utilizar la función `memoize` para gestionar de forma automática la actualización de las credenciales cuando estas caducan. Consulte el código siguiente para ver un ejemplo.

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

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

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

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

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

Para obtener más información sobre `'assume_role_params'`, consulte [AssumeRole](#).

Encadenamiento de proveedores

Es posible encadenar los proveedores de credenciales mediante la función `Aws\Credentials\CredentialProvider::chain()`. Esta función acepta un número de argumentos variádico, cada uno de los cuales es una función de proveedor de credenciales. A continuación, esta función devuelve una función nueva que se compone de las funciones proporcionadas, de modo que se invocan una tras otra hasta que uno de los proveedores devuelve una promesa que se cumple correctamente.

El `defaultProvider` utiliza esta composición para comprobar varios proveedores antes de que se produzca un error. El origen del `defaultProvider` demuestra el uso de la función `chain`.

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

Creación de un proveedor personalizado

Los proveedores de credenciales son simplemente funciones que cuando se invocan devuelven una promesa (`GuzzleHttp\Promise\PromiseInterface`) que se cumple con un objeto `Aws\Credentials\CredentialsInterface` o se rechaza con una `Aws\Exception\CredentialsException`.

Una práctica recomendada para la creación de proveedores consiste en crear una función que se invoca para crear el proveedor de credenciales real. Por ejemplo esto es el origen del proveedor `env` (ligeramente modificado para los fines del ejemplo). Observe que es una función que devuelve la función de proveedor real. Esto le permite crear fácilmente proveedores de credenciales y pasarlos como valores.

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

Proveedor de `defaultProvider`

`Aws\Credentials\CredentialProvider::defaultProvider` es el proveedor de credenciales predeterminado. Este proveedor se utiliza si se omite una opción `credentials` al crear un cliente. Primero intenta cargar las credenciales a partir de las variables de entorno, después desde un archivo `.ini` (primero un archivo `.aws/credentials` y luego un archivo `.aws/config`) y, por último, desde un perfil de instancia (primero `EcsCredentials`, seguido de los metadatos de `Ec2`).

Note

El resultado del proveedor predeterminado se memoiza automáticamente.

Proveedor de `ecsCredentials`

`Aws\Credentials\CredentialProvider::ecsCredentials` intenta cargar las credenciales mediante una solicitud GET cuyo URI se especifica mediante la variable de entorno `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` del contenedor.

```
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
]);
```

Proveedor `env`

`Aws\Credentials\CredentialProvider::env` intenta cargar las credenciales desde las variables de entorno.

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

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

Asumir el rol con un proveedor de identidad web

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider` intenta cargar las credenciales al asumir un rol. Si las variables de entorno `AWS_ROLE_ARN` y `AWS_WEB_IDENTITY_TOKEN_FILE` están presentes, el proveedor intentará asumir el rol especificado en `AWS_ROLE_ARN` utilizando el token en el disco en la ruta completa especificada en `AWS_WEB_IDENTITY_TOKEN_FILE`. Si se utilizan variables de entorno, el proveedor intentará establecer la sesión desde la variable de entorno `AWS_ROLE_SESSION_NAME`.

Si no se configuran las variables de entorno, el proveedor utilizará el perfil predeterminado o el configurado como `AWS_PROFILE`. El proveedor lee los perfiles de `~/.aws/credentials` y `~/.aws/config` de forma predeterminada y puede leer desde los perfiles especificados en la opción de configuración `filename`. El proveedor asumirá el rol en el `role_arn` del perfil al leer un token desde la ruta completa establecida en el `web_identity_token_file`. El `role_session_name` se utilizará si está establecido en el perfil.

El proveedor se denomina como parte de la cadena predeterminada y se le puede llamar directamente.

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

```
$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider();  
// Cache the results in a memoize function to avoid loading and parsing  
// the ini file on every API operation  
$provider = CredentialProvider::memoize($provider);  
  
$client = new S3Client([  
    'region'      => 'us-west-2',  
    'version'     => '2006-03-01',  
    'credentials' => $provider  
]);
```

De forma predeterminada, este proveedor de credenciales heredará la región configurada que utilizará StsClient para asumir el rol. De forma opcional, se puede proporcionar un StsClient completo. Las credenciales deben establecerse como false en cualquier StsClient proporcionada.

```
use Aws\Credentials\CredentialProvider;  
use Aws\S3\S3Client;  
use Aws\Sts\StsClient;  
  
$stsClient = new StsClient([  
    'region'      => 'us-west-2',  
    'version'     => 'latest',  
    'credentials' => false  
]);  
  
$provider = CredentialProvider::assumeRoleWithWebIdentityCredentialProvider([  
    'stsClient' => $stsClient  
]);  
// Cache the results in a memoize function to avoid loading and parsing  
// the ini file on every API operation  
$provider = CredentialProvider::memoize($provider);  
  
$client = new S3Client([  
    'region'      => 'us-west-2',  
    'version'     => '2006-03-01',  
    'credentials' => $provider  
]);
```

Proveedor ini

Aws\Credentials\CredentialProvider::ini intenta cargar las credenciales desde un [archivo de credenciales ini](#) (p. 46). De forma predeterminada, el SDK intenta cargar el perfil "predeterminado" desde un archivo ubicado en ~/.aws/credentials.

```
use Aws\Credentials\CredentialProvider;  
use Aws\S3\S3Client;  
  
$provider = CredentialProvider::ini();  
// Cache the results in a memoize function to avoid loading and parsing  
// the ini file on every API operation  
$provider = CredentialProvider::memoize($provider);  
  
$client = new S3Client([  
    'region'      => 'us-west-2',  
    'version'     => '2006-03-01',  
    'credentials' => $provider  
]);
```

Puede utilizar un perfil personalizado o una ubicación de archivo.ini proporcionando argumentos a la función que crea el proveedor.

```
$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
]);
```

Proveedor process

`Aws\Credentials\CredentialProvider::process` intenta cargar las credenciales desde un `credential_process` especificado en un [archivo ini de credenciales](#) (p. 46). De forma predeterminada, el SDK intenta cargar el perfil "predeterminado" desde un archivo ubicado en `~/.aws/credentials`. El SDK llamará al comando `credential_process` exactamente como se indica y, a continuación, leerá los datos JSON desde stdout. El comando `credential_process` debe escribir las credenciales en stdout con el formato siguiente:

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

`SessionToken` y `Expiration` son opcionales. Si se utilizan, las credenciales se considerarán temporales.

```
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
]);
```

Puede utilizar un perfil personalizado o una ubicación de archivo.ini proporcionando argumentos a la función que crea el proveedor.

```
$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
]);
```



```
]);
```

Proveedor de `instanceProfile`

`Aws\Credentials\CredentialProvider::instanceProfile` intenta cargar las credenciales desde perfiles de instancias Amazon EC2.

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

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

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

De forma predeterminada, el proveedor vuelve a intentar buscar licencias hasta tres veces. El número de reintentos se puede establecer con la opción `retries` y se puede deshabilitar por completo al establecer la opción en 0.

```
use Aws\Credentials\CredentialProvider;

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

Note

Puede deshabilitar este intento de cargar desde perfiles de instancias Amazon EC2 estableciendo la variable de entorno `AWS_EC2_METADATA_DISABLED` en `true`.

Memoización de credenciales

A veces es necesario crear un proveedor de credenciales que recuerde el valor de retorno anterior. Esto puede resultar útil para el rendimiento cuando cargar credenciales es una operación costosa o cuando se utiliza la clase `Aws\Sdk` para compartir un proveedor de credenciales entre varios clientes. Puede añadir memoización a un proveedor de credenciales encapsulando la función de proveedor de credenciales en una función `memoize`.

```
use Aws\Credentials\CredentialProvider;

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

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

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

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

Cuando vencen las credenciales memoizadas, el encapsulador de memoización invoca al proveedor encapsulado para actualizarlas.

Uso de credenciales temporales desde AWS STS

[AWS Security Token Service](#) (AWS STS) le permite solicitar privilegios limitados, credenciales temporales para los usuarios de AWS IAM o para los usuarios que se autenticuen a través de la identidad federada.

Un caso de uso común de las credenciales temporales consiste en conceder a las aplicaciones móviles o del lado del cliente acceso a los recursos de AWS autenticando a los usuarios a través de proveedores de identidad externos (consulte [Identidades web federadas](#)).

Note

Las credenciales temporales que AWS STS genera no son compatibles con todos los servicios. Para saber si el servicio que utiliza es compatible con el uso de credenciales temporales, consulte [Credenciales de seguridad temporales de IAM](#).

Obtención de credenciales temporales

AWS STS tiene varias operaciones que devuelven credenciales temporales, pero la operación `GetSessionToken` es la más sencilla que se puede demostrar. Suponiendo que tiene una instancia de `Aws\Sts\StsClient` almacenada en la variable `$stsClient`, puede llamarla como se indica a continuación.

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

El resultado para `GetSessionToken` y las operaciones AWS STS restantes siempre contiene un valor `'Credentials'`. Si imprime el resultado (por ejemplo, `print_r($result)`), tendrá un aspecto similar al siguiente.

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

Proporcionar credenciales temporales al AWS SDK for PHP

Puede utilizar credenciales temporales con otro cliente de AWS creando una instancia del cliente y transfiriendo los valores que ha recibido desde AWS STS directamente.

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

También puede crear un objeto `Aws\Credentials\Credentials` y utilizarlo cuando cree una instancia del cliente.

```
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
]);
```

Sin embargo, la mejor manera de proporcionar credenciales temporales es utilizar el método de ayudante `createCredentials()` incluido con `StsClient`. Este método extrae los datos de un resultado de AWS STS y crea el objeto `Credentials` para usted.

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

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

Para obtener más información acerca de por qué es posible que necesite utilizar credenciales temporales en su aplicación o proyecto, consulte [Escenarios para conceder acceso temporal](#) en la documentación de AWS STS.

Uso de credenciales no modificables

Al probar nuevos servicios o depurar errores, los desarrolladores a menudo desean incluir las credenciales de AWS cuando crean el cliente. Consulte lo siguiente para obtener un ejemplo de cómo autenticar en AWS, pero hágalo con precaución. En las [credenciales para AWS SDK para PHP versión 3 \(p. 44\)](#) se incluyen muchas formas recomendadas para añadir credenciales a su proyecto de forma segura.

Warning

Codificar de forma rígida sus credenciales puede ser peligroso, ya que es fácil confirmar accidentalmente las credenciales en un repositorio de SCM. Agregar credenciales directamente

en su código de producción puede exponer potencialmente sus credenciales a más personas de las que desea. También puede dificultar la rotación de credenciales en el futuro.

Si decide utilizar credenciales codificadas de forma rígida en un cliente de SDK, proporcione una matriz asociativa de pares de clave y valor "clave", "secreto" y "token" opcional a la opción "credenciales" de un constructor de cliente.

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

Creación de clientes anónimos

En algunos casos es posible que desee crear un cliente que no esté asociado a ninguna credencial. De este modo podrá realizar solicitudes anónimas a un servicio.

Por ejemplo, puede configurar tanto objetos de Amazon S3 como dominios de Amazon CloudSearch para que permitan el acceso anónimo.

Para crear un cliente anónimo, debe establecer la opción 'credentials' en false.

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

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

Objetos comando en el AWS SDK for PHP versión 3

AWS SDK for PHP usa el [patrón de comandos](#) para encapsular los parámetros y el controlador que se utilizarán para transferir una solicitud HTTP en un momento posterior.

Uso implícito de comandos

Si examina cualquier clase de cliente, verá que los métodos que corresponden a las operaciones de la API no existen en realidad. Se implementan mediante el método mágico `__call()`. Estos pseudométodos son en realidad accesos directos que encapsulan el uso de objetos comando por parte del SDK.

Normalmente no tendrá que interactuar con objetos comando directamente. Cuando llama a métodos como `Aws\S3\S3Client::putObject()`, el SDK en realidad crea un objeto `Aws`

\CommandInterface utilizando los parámetros proporcionados, ejecuta el comando y devuelve un objeto `Aws\ResultInterface` con la información devuelta (o genera una excepción en caso de error). El flujo es similar cuando se llama a cualquiera de los métodos `Async` de un cliente (por ejemplo, `Aws\S3\S3Client::putObjectAsync()`): el cliente crea un comando basado en los parámetros proporcionados, serializa una solicitud HTTP, inicia la solicitud y devuelve una promesa.

Los siguientes ejemplos son funcionalmente equivalentes.

```
$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);
```

Parámetros de comando

Todos los comandos admiten algunos parámetros especiales que no forman parte de una API del servicio, sino que controlan el comportamiento del SDK.

@http

Con este parámetro se puede ajustar la forma en que el controlador HTTP subyacente ejecuta la solicitud. Las opciones que puede incluir en el parámetro `@http` son las mismas que puede establecer al crear instancias para el cliente con la [opción de cliente "http"](#) (p. 32).

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

@retries

Al igual que la [opción del cliente "retries"](#) (p. 39), `@retries` controla cuántas veces un comando se pueden reintentar antes de que se considere que se ha producido un error. Si establece el valor 0, se deshabilitan los reintentos.

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

Note

Cuando se deshabilitan los reintentos en un cliente, no es posible habilitarlos de manera selectiva en los comandos individuales que se pasan a ese cliente.

Crear objetos comando

Puede crear un comando con el método `getCommand()` de un cliente. No ejecuta ni transfiere inmediatamente una solicitud HTTP, solo se ejecuta cuando se especifica en el método `execute()` del cliente. Esto le da la oportunidad de modificar el objeto comando antes de ejecutarlo.

```
$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);
```

Comando HandlerList

Cuando se crea un comando desde un cliente, se le asigna un clon del objeto `Aws\HandlerList` del cliente. Se asigna al comando un clon de la lista de controladores del cliente para permitirle usar middleware y controladores personalizados sin afectar a los demás comandos que ejecuta el cliente.

Esto significa que puede utilizar un cliente HTTP diferente para cada comando (por ejemplo, `Aws\MockHandler`) y añadir un comportamiento personalizado para cada comando mediante middleware. En el ejemplo siguiente se usa `MockHandler` para crear resultados simulados en lugar de enviar solicitudes HTTP reales.

```
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'
```

Además de cambiar el controlador que usa el comando, también puede inyectarle middleware personalizado. En el ejemplo siguiente se usa el middleware `tap`, que actúa como observador en la lista de controladores.

```
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

El objeto `Aws\CommandPool` permite ejecutar comandos de forma simultánea mediante un iterador que proporciona objetos `Aws\CommandInterface`. `CommandPool` garantiza la ejecución simultánea de un número constante de comandos mientras se itera por los comandos del grupo (cuando se completan comandos, se ejecutan otros para asegurar el número constante).

El siguiente es un ejemplo muy sencillo de cómo enviar algunos comandos con `CommandPool`.

```
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();
```

Este ejemplo no funciona bien con `CommandPool`. Vamos a probar un ejemplo más complejo. Supongamos que desea cargar archivos del disco en un bucket de Amazon S3. Para obtener una lista de archivos del disco, podemos usar `DirectoryIterator` de PHP. Este iterador devuelve objetos `SplFileInfo`. `CommandPool` acepta un iterador que devuelve objetos `Aws\CommandInterface`, por lo que asignamos los objetos `SplFileInfo` para conseguir objetos `Aws\CommandInterface`.

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

use Aws\Exception\AwsException;
use Aws\S3\S3Client;
use Aws\CommandPool;
use Aws\CommandInterface;
use Aws\ResultInterface;
```

```

use GuzzleHttp\Promise\PromiseInterface;

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

$fromDir = '/path/to/dir';
$toBucket = '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"; });
```

Configuración de `CommandPool`

El constructor `Aws\CommandPool` acepta distintas opciones de configuración.

`concurrency` (invocable|entero)

Número máximo de comandos que se ejecutan de forma simultánea. Especifique una función para cambiar el tamaño del grupo de forma dinámica. La función recibe el número actual de solicitudes pendientes y se espera que devuelva un número entero que representa el nuevo límite de tamaño del grupo.

`before` (invocable)

Function que se invoca antes de enviar cada comando. La función `before` acepta el comando y la clave del iterador del comando. Puede cambiar el comando de la `before` como sea necesario antes de enviar el comando.

`fulfilled` (invocable)

Función que se invoca invocar cuando una promesa se ha cumplido. La función recibe el objeto resultado, el ID del iterador del que proviene el resultado y la promesa agregada que se puede resolver o rechazar si es necesario cerrar el grupo.

`rejected` (invocable)

Función que se invoca invocar cuando una promesa se ha rechazado. La función recibe un objeto `Aws\Exception`, el ID del iterador del que proviene la excepción y la promesa agregada que se puede resolver o rechazar si es necesario cerrar el grupo.

Recopilación manual de elementos no utilizados entre comandos

Si se alcanza el límite de memoria al utilizar grupos de comandos de gran tamaño, puede deberse a las referencias cíclicas generadas por el SDK que el [recolector de elementos no utilizados de PHP](#) todavía no había recopilado cuando se alcanzó el límite de memoria. Si se invoca manualmente el algoritmo de recopilación entre los comandos, es posible que los ciclos se recopilen antes de alcanzar dicho límite. En el siguiente ejemplo, se crea un `CommandPool` que invoca el algoritmo de recopilación mediante una devolución de llamada antes del envío de cada de parte. Tenga en cuenta que invocar el recolector de elementos no utilizados conlleva un costo de rendimiento y su uso óptimo dependerá de su caso de uso y su entorno.

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

Promesas en el AWS SDK for PHP versión 3

AWS SDK for PHP usa promesas para permitir los flujos de trabajo asíncronos y este carácter asíncrono permite enviar solicitudes HTTP simultáneas. La especificación de promesa que usa el SDK es [Promises/A+](#).

¿Qué es una promesa?

Una promesa representa el resultado final de una operación asíncrona. La forma principal de interactuar con una promesa es mediante su método `then`. Este método registra las devoluciones de llamadas para recibir el valor final de la promesa o la razón por la que no se puede cumplir.

AWS SDK for PHP utiliza el paquete Composer [guzzlehttp/promete](#) en la implementación de sus promesas. Las promesas de Guzzle admiten flujos de trabajo con y sin bloqueo, y pueden usarse en cualquier bucle de eventos sin bloqueo.

Note

Las solicitudes HTTP se envían de forma simultánea en AWS SDK for PHP mediante un solo subproceso, en el que se usan llamadas sin bloqueo para transferir una o varias solicitudes HTTP, mientras que se reacciona a cambios de estado (por ejemplo, cuando se cumplen o incumplen promesas).

Promesas en el SDK

Las promesas se utilizan en todo el SDK. Por ejemplo, las promesas se utilizan en la mayoría de las abstracciones de alto nivel que proporciona el SDK: [paginadores](#) (p. 83), [esperadores](#) (p. 85), [grupos de comandos](#) (p. 64), [cargas multiparte](#) (p. 233), [transferencias de directorio/bucket de S3](#) (p. 113), etc.

Todos los clientes que proporciona el SDK devuelven promesas cuando se invoca cualquiera de los métodos con el sufijo `Async`. Por ejemplo, el siguiente código muestra cómo crear una promesa para obtener los resultados de una operación `DescribeTable` de Amazon DynamoDB.

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

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

Observe que puede llamar a `describeTable` o `describeTableAsync`. Estos métodos son métodos `__call` mágicos en un cliente basados en el modelo de API y el número `version` asociados con el cliente. Con métodos como `describeTable` sin el sufijo `Async`, el cliente se bloqueará mientras envía una solicitud HTTP y devolverá un objeto `Aws\ResultInterface` o generará una excepción `Aws\Exception\AwsException`. El sufijo al nombre de operación con `Async` (es decir, `describeTableAsync`) significa que el cliente creará una promesa que finalmente se cumple con un objeto `Aws\ResultInterface` o se rechaza con una excepción `Aws\Exception\AwsException`.

Important

Cuando se devuelve la promesa, puede que el resultado ya haya llegado (por ejemplo, cuando se utiliza un controlador simulado) o puede que la solicitud HTTP no se han iniciado.

Puede registrar una devolución de llamada con la promesa usando el método `then`. Este método acepta dos devoluciones de llamada: `$onFulfilled` y `$onRejected`, ambas opcionales. La devolución de llamada `$onFulfilled` se invoca si la promesa se cumple, y la devolución de llamada `$onRejected` se invoca si la promesa se rechaza (lo que significa que ha fallado).

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

```
    },  
    function ($reason) {  
        echo "The promise was rejected with {$reason}";  
    }  
);
```

Ejecutar comandos simultáneamente

Es posible agrupar múltiples promesas para que se ejecuten de forma simultánea. Esto puede lograrse integrando el SDK con un bucle de eventos sin bloqueo, o creando múltiples promesas y esperando a que se completen de forma simultánea.

```
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

El [CommandPool](#) (p. 64) proporciona un mecanismo más potente para ejecutar varias operaciones de la API de forma simultánea.

Encadenar promesas

Uno de los aspectos más interesantes de las promesas es pueden combinarse, lo que le permite crear canalizaciones de transformación. Las promesas se combinan encadenando devoluciones de llamada `then` con devoluciones de llamada `then` sucesivas. El valor que devuelve un método `then` es una promesa cumplida o rechazada en función del resultado de las devoluciones de llamadas indicadas.

```
$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

El valor que devuelve una devolución de llamada de una promesa es el argumento `$value` que se suministra a promesas posteriores. Si desea especificar un valor para las promesas posteriores en la cadena, debe devolver un valor en la función de devolución de llamada.

Reenviar rechazos

Es posible registrar una devolución de llamada cuando se rechaza una promesa. Si cualquier devolución de llamada contiene una excepción, la promesa se rechaza con esa excepción, al igual que la siguiente promesa de la cadena. Si devuelve un valor correcto desde una devolución de llamada `$onRejected` la siguiente promesa de la cadena se cumplirá con el valor de retorno de la devolución de llamada `$onRejected`.

Esperar a las promesas

Para forzar que se complete una promesa de forma síncrona, puede usar su método `wait`.

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

Si se encuentra una excepción al llamar a la función `wait` de una promesa, la promesa se rechaza con esa excepción y se genera una excepción.

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

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

Cuando se invoca `wait` para una promesa que se ya se ha cumplido, no se activa la espera. Simplemente se devuelve el valor generado previamente.

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

Cuando se invoca `wait` para una promesa que se ha rechazado, se genera una excepción. Si el motivo de rechazo es una instancia de `\Exception`, se genera el motivo como excepción. De lo contrario, se

genera `GuzzleHttp\Promise\RejectionException` y el motivo puede obtenerse llamando al método `getReason` de la excepción.

Note

Las llamadas a operaciones API de AWS SDK for PHP se rechazan con subclases de la clase `Aws\Exception\AwsException`. Sin embargo, es posible que la razón indicada al método `then` sea diferente, ya que el uso del middleware personalizado modifica el motivo del rechazo.

Cancelar promesas

Las promesas se puede cancelar utilizando su método `cancel()`. Si una promesa ya se ha resuelto, llamar a `cancel()` no tendrá ningún efecto. La cancelación de una promesa cancela la promesa y cualesquiera otras que estén esperando su resultado. Las promesas canceladas se rechazan con `GuzzleHttp\Promise\RejectionException`.

Combinar promesas

Puede combinar promesas en grupos con los que crear flujos de trabajo más complejos. El paquete `guzzlehttp/promise` contiene distintas funciones que puede utilizar para combinar promesas.

Encontrará la documentación de la API con todas las funciones para combinar promesas en [namespace-GuzzleHttp.Promise](#).

each y each_limit

Utilice la [CommandPool](#) (p. 64) cuando tenga una cola de tareas de comandos `Aws\CommandInterface` para realizar de forma simultánea con un tamaño de grupo fijo (los comandos pueden estar en memoria o ser generados por un iterador perezoso). `CommandPool` garantiza que un número fijo de comandos se envían de forma simultánea hasta que se agote el iterador.

`CommandPool` solo funciona con comandos que ejecuta un mismo cliente. Puede usar la función `GuzzleHttp\Promise\each_limit` para enviar comandos de diferentes clientes de forma simultánea con un número fijo de comandos.

```
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();
```

Corutinas de promesas

Una de las características con más posibilidades de la biblioteca de promesas Guzzle es que permite utilizar corutinas de promesas que permiten escribir flujos de trabajo asíncronos de forma similar a los flujos de trabajo síncronos tradicionales. De hecho, AWS SDK for PHP utiliza promesas corutinas en la mayoría de las abstracciones de alto nivel.

Supongamos que desea crear varios buckets y cargar un archivo en un bucket cuando esté disponible, y que quiere ejecutar todas estas operaciones de forma simultánea para que se hagan lo más rápido posible. Es fácil conseguirlo combinando varias promesas corutinas mediante la función `all()`.

```
use GuzzleHttp\Promise;

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

// Create the following buckets
$buckets = ['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();
```

Controladores y middleware en el AWS SDK for PHP versión 3

El mecanismo principal para ampliar AWS SDK for PHP es mediante controladores y middleware. Cada clase del cliente del SDK posee una instancia `Aws\HandlerList` a la que se puede obtener acceso utilizando el método `getHandlerList()` de un cliente. Puede recuperar la `HandlerList` de un cliente y modificarla para añadir o eliminar el comportamiento del cliente.

Handlers

Un controlador es una función que realiza la transformación real de un comando y una solicitud en un resultado. Un controlador suele enviar solicitudes HTTP. Los controladores pueden incluir middleware para

aumentar su comportamiento. Un controlador es una función que acepta una `Aws\CommandInterface` y una `Psr\Http\Message\RequestInterface` y devuelve una promesa que se cumple con una `Aws\ResultInterface` o se rechaza con un motivo `Aws\Exception\AwsException`.

A continuación se presenta un controlador que devuelve el mismo resultado simulado para cada llamada.

```
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);
};
```

A continuación, puede utilizar este controlador con un cliente del SDK proporcionando una opción `handler` en el constructor de un cliente.

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

También puede cambiar el controlador de un cliente después de crearlo utilizando el método `setHandler` de una `Aws\ClientInterface`.

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

Controlador simulado

Le recomendamos que utilice el `MockHandler` para escribir pruebas que utilicen el SDK. Puede utilizar el parámetro `Aws\MockHandler` para devolver los resultados simulados o lanzar excepciones simuladas. El usuario debe poner los resultados o las excepciones en la cola, y el `MockHandler` los saca de la cola siguiendo el orden FIFO.

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

$mock = new MockHandler();

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

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

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-west-2',
```

```
'version' => 'latest',  
'handler' => $mock  
]);  
  
// Result object response will contain ['foo' => 'bar']  
$result = $client->listTables();  
  
// This will throw the exception that was enqueued  
$client->listTables();
```

Middleware

El middleware es un tipo especial de función de alto nivel que aumenta el comportamiento de la transferencia de un comando y lo delega al "siguiente" controlador. Las funciones de middleware aceptan una `Aws\CommandInterface` y una `Psr\Http\Message\RequestInterface` y devuelven una promesa que se cumple con una `Aws\ResultInterface` o se rechaza con un motivo `Aws\Exception\AwsException`.

Un middleware es una función de orden superior que modifica un comando, solicitud o resultado al pasar por el middleware. Un middleware tiene el siguiente aspecto.

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

Un middleware recibe un comando a ejecutar y un objeto Request opcional. El middleware puede elegir aumentar la solicitud y el comando o dejarlas tal y como están. A continuación, un middleware invoca al siguiente controlador de la cadena o puede optar por cortocircuitar el siguiente controlador y devolver una promesa. La promesa que se crea invocando el siguiente controlador también se puede aumentar utilizando el método `then` de la promesa para modificar el posible resultado o error antes de devolver la promesa a la pila de middleware.

HandlerList

El SDK utiliza una `Aws\HandlerList` para administrar el middleware y los controladores utilizados al ejecutar un comando. Cada cliente del SDK posee una `HandlerList` y dicha `HandlerList` se clona y se agrega a cada comando que crea un cliente. Puede asociar un middleware y el controlador predeterminado que se va a utilizar para cada comando creado por un cliente añadiendo un middleware a la `HandlerList` del cliente. Puede añadir y eliminar middleware de comandos específicos modificando el `HandlerList` propiedad de un comando específico.

Una `HandlerList` representa una pila de middleware que se utiliza para encapsular un controlador. Para ayudar a administrar la lista de middleware y el orden en que encapsulan un controlador, la `HandlerList`

divide la pila en pasos denominados que representan partes del ciclo de vida de la transferencia de un comando:

1. **init**: añade parámetros predeterminados
2. **validate**: valida los parámetros obligatorios
3. **build**: serializa una solicitud HTTP para enviarla
4. **sign**: firma la solicitud HTTP serializada
5. **<controlador>** (no es un paso pero ejecuta la transferencia real)

init

Este paso del ciclo de vida representa la inicialización de un comando. Aún no se ha serializado ninguna solicitud. Este paso se suele utilizar para añadir los parámetros predeterminados a un comando.

Puede añadir un middleware al paso **init** con los métodos `appendInit` y `prependInit`, donde `appendInit` añade el middleware al final de la lista `prepend` mientras que `prependInit` añade el middleware al principio de la lista `prepend`.

```
use Aws\Middleware;

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

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

validar

Este paso del ciclo de vida se utiliza para validar los parámetros de entrada de un comando.

Puede añadir un middleware al paso **validate** con los métodos `appendValidate` y `prependValidate`, donde `appendValidate` añade el middleware al final de la lista `validate` mientras que `prependValidate` añade el middleware al principio de la lista `validate`.

```
use Aws\Middleware;

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

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

build

Este paso del ciclo de vida se utiliza para serializar una solicitud HTTP para el comando que se está ejecutando. Los eventos del ciclo de vida posteriores recibirán un comando y una solicitud HTTP PSR-7.

Puede añadir un middleware al paso **build** con los métodos `appendBuild` y `prependBuild`, donde `appendBuild` añade el middleware al final de la lista `build` mientras que `prependBuild` añade el middleware al principio de la lista `build`.

```
use Aws\Middleware;

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

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

sign

Este paso del ciclo de vida se suele utilizar para firmar solicitudes HTTP antes de que se envíen a través de la red. Debe abstenerse de realizar cambios en una solicitud HTTP cuando ya esté firmada para evitar errores de firma.

Este es el último paso de la `HandlerList` antes de que un controlador transfiera la solicitud HTTP.

Puede añadir un middleware al paso `sign` con los métodos `appendSign` y `prependSign`, donde `appendSign` añade el middleware al final de la lista `sign` mientras que `prependSign` añade el middleware al principio de la lista `sign`.

```
use Aws\Middleware;

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

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

Middleware disponible

El SDK proporciona varios middleware que puede utilizar para aumentar el comportamiento de un cliente o para observar la ejecución de un comando.

mapCommand

El middleware `Aws\Middleware::mapCommand` resulta útil cuando hay que modificar un comando antes de que se serialice como solicitud HTTP. Por ejemplo, `mapCommand` se puede utilizar para llevar a cabo una validación o para añadir parámetros predeterminados. La función `mapCommand` acepta una función invocable que admite un objeto `Aws\CommandInterface` y devuelve un objeto `Aws\CommandInterface`.

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

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

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

```
        // Be sure to return the command!
        return $command;
    }},
    'add-param'
);
```

mapRequest

El middleware `Aws\Middleware::mapRequest` resulta útil cuando hay que modificar una solicitud después de serializarla pero antes de enviarla. Por ejemplo, se puede utilizar para añadir encabezados HTTP personalizados a una solicitud. La función `mapRequest` acepta una función invocable que admite un argumento `Psr\Http\Message\RequestInterface` y devuelve un objeto `Psr\Http\Message\RequestInterface`.

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

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

Ahora al ejecutar el comando, se envía con el encabezado personalizado.

Important

Tenga en cuenta que el middleware se adjuntó a la lista de controladores al final del paso `build`. De este modo, se garantiza que se cree una solicitud antes de invocar este middleware.

mapResult

El middleware `Aws\Middleware::mapResult` resulta útil para modificar el resultado de una ejecución del comando. La función `mapResult` acepta una función invocable que admite un argumento `Aws\ResultInterface` y devuelve un objeto `Aws\ResultInterface`.

```
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;
    })
);
```

Ahora cuando se ejecute el comando, el resultado incluirá un atributo `foo`.

history

El middleware `history` es útil para probar que el SDK ejecuta los comandos previstos, envía las solicitudes HTTP esperadas y recibe los resultados previstos. Se trata básicamente de un middleware que actúa de forma similar al historial de un navegador web.

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

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

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

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

Un contenedor de historiales `Aws\History` almacena 10 entradas de forma predeterminada antes de purgar las entradas. Puede personalizar el número de entradas transfiriendo el número de entradas que desea conservar en el constructor.

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

Puede examinar el contenedor de historiales después de ejecutar las solicitudes que transfieren el middleware del historial.

```
// 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

El middleware `tap` se utiliza como observador. Puede utilizar este middleware para invocar a funciones al enviar los comandos a través de la cadena de middleware. La función `tap` acepta una función invocable que admite la `Aws\CommandInterface` y una `Psr\Http\Message\RequestInterface` opcional que se está ejecutando.

```
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: ' . $req->getMethod() . "\n";
        }
    })
);
```

Crear controladores personalizados

Un controlador es simplemente una función que acepta un objeto `Aws\CommandInterface` y un objeto `Psr\Http\Message\RequestInterface`, y que devuelve una `GuzzleHttp\Promise\PromiseInterface` que se cumple con una `Aws\ResultInterface` o se rechaza con una `Aws\Exception\AwsException`.

Aunque el SDK tiene varias opciones `@http`, un controlador solo necesita saber cómo utilizar las siguientes opciones:

- [connect_timeout](#) (p. 32)
- [debug](#) (p. 33)
- [decode_content](#) (p. 33) (opcional)
- [delay](#) (p. 34)
- [progress](#) (p. 34) (opcional)
- [proxy](#) (p. 35)
- [sink](#) (p. 35)
- [synchronous](#) (p. 36) (opcional)
- [stream](#) (p. 36) (opcional)
- [timeout](#) (p. 36)
- [verify](#) (p. 37)
- [http_stats_receiver](#) (opcional): es una función para invocar con una matriz asociativa de estadísticas de transferencia HTTP si se solicita utilizando el parámetro de configuración [stats](#) (p. 28).

A menos que la opción se especifique como opcional, un controlador DEBE controlar la opción o devolver una promesa rechazada.

Además de la gestión de opciones `@http` específicas, un controlador DEBE añadir un encabezado `User-Agent` que tiene el siguiente formato, donde "3.X" se puede reemplazar por `Aws\Sdk::VERSION` y "HandlerSpecificData/version..." debe sustituirse por la cadena `User-Agent` específica del controlador.

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

Flujos en el AWS SDK for PHP versión 3

Como parte de su integración del estándar de mensajes HTTP [PSR-7](#), AWS SDK for PHP utiliza [PSR-7StreamInterface](#) internamente como su abstracción en los [flujos de PHP](#). Cualquier comando que disponga de un campo de entrada definido como `blob`, como el parámetro `Body` en un [comando S3::PutObject](#), puede cumplirse con una cadena, un recurso de flujo de PHP o una instancia de `Psr\Http\Message\StreamInterface`.

Warning

El SDK se apropia de cualquier recurso de flujo de PHP sin procesar suministrado como parámetro de entrada a un comando. El flujo se consume y cierra en su nombre. Si necesita compartir un flujo entre una operación del SDK y su código, encapsúlelo en una instancia de `GuzzleHttp\Psr7\Stream` antes de incluirlo como un parámetro del comando. El SDK consume el flujo, por lo que su código debe tener en cuenta el movimiento del cursor interno del flujo. Los flujos de Guzzle llaman a `fclose` en el recurso de flujo subyacente cuando el recolector de elementos no utilizados de PHP los destruye, por lo que no es necesario que cierre el flujo usted mismo.

Decoradores de flujo

Guzzle proporciona varios decoradores de flujo que puede utilizar para controlar la forma en que el SDK y Guzzle interactúan con el recurso de flujo suministrado como parámetro de entrada a un comando. Estos decoradores pueden modificar la forma en que los controladores leen y buscan un flujo determinado. A continuación se muestra una lista parcial; encontrará más información en el [repositorio de GuzzleHttpPsr7](#).

AppendStream

[GuzzleHttp\AppendStream](#)

Lee varios flujos, uno detrás del otro.

```
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](#)

Se utiliza para permitir la búsqueda de bytes leídos previamente en flujos no rastreables. Esto puede resultar útil cuando la transferencia de un cuerpo de entidad no rastreable falla debido a la necesidad de rebobinar el flujo (por ejemplo, debido a un redireccionamiento). Los datos que se leen en el flujo remoto se almacenan en búfer en un flujo temporal de PHP, de modo que los bytes leídos previamente se almacenan primero en caché en la memoria y luego en el disco.

```
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\flujo de\Psr7\InflateStream

Utiliza el filtro `zlib.inflate` de PHP para inflar o desinflar contenido comprimido en un zip.

Este decorador de flujos omite los primeros 10 bytes del flujo dado para eliminar el encabezado gzip, convierte el flujo proporcionado en un recurso de flujo de PHP y, a continuación, añade el filtro `zlib.inflate`. El flujo se vuelve a convertir en un recurso de flujo de Guzzle que se va a utilizar como flujo de Guzzle.

LazyOpenStream

GuzzleHttp: \Psr7\LazyOpenStream

Lee o escribe lentamente en un archivo que se abre después de que se produzca una operación de E/S en el flujo.

```
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

GuzzleHttpTransmisiones\Psr7\LimitStream

Se utiliza para leer un subconjunto o sector de un objeto de flujo existente. Esto puede resultar útil para dividir un archivo de gran tamaño en partes más pequeñas que se envían en fragmentos (por ejemplo, la API de carga multiparte de Amazon S3).

```
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\Ps7\NoSeekStream](#)

Encapsula un flujo y no permite que se encuentre.

```
use GuzzleHttp\Ps7;  
  
$original = Ps7\stream_for('foo');  
$noSeek = new Ps7\NoSeekStream($original);  
  
echo $noSeek->read(3);  
// foo  
var_export($noSeek->isSeekable());  
// false  
$noSeek->seek(0);  
var_export($noSeek->read(3));  
// NULL
```

PumpStream

[GuzzleHttp\Ps7\PumpStream](#)

Proporciona un flujo de solo lectura que bombea datos desde una función invocable de PHP.

A la hora de invocar la función invocable proporcionada, la PumpStream pasa la cantidad de datos solicitados a leer a la función invocable. La función invocable puede elegir omitir este valor y devolver más o menos bytes de los solicitados. Los datos adicionales devueltos por la función invocable proporcionada se almacenan en búfer internamente hasta que se vacíe utilizando la función read() del PumpStream. La función invocable proporcionada DEBE devolver el valor false cuando no hay más datos que leer.

Implementar decoradores de flujo

Crear un decorador de flujos es un saludo muy fácil a [GuzzleHttp\Ps7\StreamDecoratorTrait](#). Esta característica proporciona métodos que implementan `Ps7\Http\Message\StreamInterface` al conectar mediante proxy con un flujo subyacente. Simplemente use la característica `StreamDecoratorTrait` e implemente sus métodos personalizados.

Por ejemplo, supongamos que quiere llamar a una función específica cada vez que se lea el último byte de un flujo. Esto podría implementarse anulando el método read().

```
use Psr\Http\Message\StreamInterface;  
use GuzzleHttp\Ps7\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;
    }
}
```

Este decorador se podría agregar a cualquier flujo existente y utilizarse del siguiente modo.

```
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!"
```

Paginadores en el AWS SDK for PHP versión 3

Algunas operaciones de servicios de AWS están paginadas y responden con resultados truncados. Por ejemplo, la operación `ListObjects` de Amazon S3 solo devuelve un máximo de 1000 objetos a la vez. Las operaciones como estas (normalmente con el prefijo "lista" o "describir") requieren solicitudes posteriores con parámetros de token (o marcador) para obtener todo el conjunto de resultados.

Los paginadores de son una característica de AWS SDK for PHP que actúa como una abstracción de este proceso para facilitar a los desarrolladores el uso de APIs paginadas. Un paginador es básicamente un iterador de resultados. Se crea con el método `getPaginator()` del cliente. Cuando llama a `getPaginator()` debe proporcionar el nombre de la operación y los argumentos de la operación (del mismo modo que lo hace al ejecutar una operación). Puede iterar sobre un objeto de paginador con `foreach` para obtener objetos `Aws\Result` individuales.

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

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

Objetos de paginador

El objeto que el método `getPaginator()` devuelve es una instancia de la clase `Aws\ResultPaginator`. Esta clase implementa la interfaz `iterator` nativa de PHP, motivo por el cual funciona con `foreach`. También se puede utilizar con funciones de iterador, como `iterator_to_array`, y se integra bien con [iteradores SPL](#) como el objeto `LimitIterator`.

Los objetos del paginador solo contienen una "página" de resultados a la vez y se ejecutan lentamente. Esto significa que realizan solo tantas solicitudes como necesitan para obtener la página actual de

resultados. Por ejemplo, la operación `ListObjects` de Amazon S3 solo devuelve un máximo de 1000 objetos a la vez, por lo que si su bucket tiene ~ 10 000 objetos, el paginador debería realizar 10 solicitudes en total. Cuando itera los resultados, la primera solicitud se ejecuta cuando comienza la iteración, la segunda en la segunda iteración del bucle, y así sucesivamente.

Enumeración de datos a partir de los resultados

Los objetos del paginador tienen un método llamado `search()` que le permiten crear iteradores para datos que están en un conjunto de resultados. Cuando llame a `search()`, proporcione una [expresión JMESPath para especificar los datos que se van a extraer](#). (p. 86) Llamar a `search()` devuelve un iterador que genera los resultados de la expresión en cada página de resultados. Esto se evalúa lentamente, a medida que recorre el iterador devuelto.

El siguiente ejemplo es equivalente al ejemplo de código anterior, pero utiliza el método `ResultPaginator::search()` para ser más conciso.

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

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

Las expresiones JMESPath de le permiten hacer cosas bastante complejas. Por ejemplo, si desea imprimir todas las claves de objetos y prefijos habituales (es decir, hacer un `ls` de un bucket), puede hacer lo siguiente.

```
// 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";
}
```

Paginación asíncrona

Puede iterar a través de los resultados de un paginador de forma asíncrona proporcionando una devolución de llamada para el método `each()` de un `Aws\ResultPaginator`. La devolución de llamada se invoca para cada valor que el paginador genera.

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

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

Note

Con el método `each()` se puede paginar por los resultados de una operación de la API y enviar al mismo tiempo otras solicitudes de forma asíncrona.

La promesa basada en la corutina subyacente generará un valor de retorno no nulo a partir de la devolución de llamada. Esto significa que puede devolver promesas desde la devolución de llamada que deben resolverse antes de continuar la iteración por los elementos restantes, básicamente combinando otras promesas a la iteración. El último valor no nulo que devuelve la devolución de llamada es el resultado que responde a la promesa de todas las promesas posteriores. Si el último valor devuelto es una promesa, la resolución de esa promesa es el resultado que responde a las promesas posteriores o las rechaza.

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

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

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

Esperadores en el AWS SDK for PHP versión 3

Los esperadores facilitan el trabajo con sistemas de consistencia final proporcionando una forma abstracta de esperar hasta que un recurso pasa a un estado determinado sondeando el recurso. Puede encontrar una lista de los esperadores compatibles con un cliente en la [documentación de la API](#) para una única versión de un cliente de servicio. Para navegar allí, vaya a la página del cliente en la documentación de la API, desplácese hasta el número de versión específico (representado por una fecha) y desplácese hacia abajo hasta la sección "Waiters". [Este enlace le llevará a la sección de esperadores de S3.](#)

En el siguiente ejemplo, el cliente de Amazon S3 se utiliza para crear un bucket. A continuación, el esperador se utiliza para esperar hasta que exista el bucket.

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

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

Si el esperador tiene que sondear el bucket demasiadas veces, lanzará una excepción `\RuntimeException`.

Configuración del esperador

Los esperadores actúan en función de una matriz asociativa de opciones de configuración. Todas las opciones que utiliza un esperador determinado tienen valores predeterminados, pero se pueden reemplazar por otros para aplicar estrategias de espera distintas.

Puede modificar las opciones de configuración de los esperadores pasando una matriz asociativa de opciones `@waiter` al argumento `$args` de los métodos `waitUntil()` y `getWaiter()` de un cliente.

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

delay (entero)

Es el número de segundos de retraso entre los intentos de sondeo. Cada esperador dispone de un valor de configuración `delay` predeterminado, pero es posible que tenga que modificarlo para casos de uso específicos.

maxAttempts (int)

Es el número máximo de intentos de sondeo a emitir antes de que el esperador falle. Esta opción garantiza que no tenga que esperar un recurso de forma indefinida. Cada esperador dispone de un valor de configuración `maxAttempts` predeterminado, pero es posible que tenga que modificarlo para casos de uso específicos.

initDelay (int)

Es el intervalo de tiempo en segundos que hay que esperar hasta el primer intento de sondeo. Esto puede ser útil cuando se espera a un recurso, del que se sabe que tarda un rato en pasar al estado deseado.

before (invocable)

Es una función invocable PHP que se invoca antes de cada intento. La función invocable se invoca con el comando `Aws\CommandInterface` que está a punto de ejecutarse y el número de intentos que se han ejecutado hasta el momento. La función invocable `before` se puede utilizar para modificar los comandos antes de ejecutarlos o para proporcionar información de progreso.

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

Espera asíncrona

Además de la espera síncrona, puede invocar un esperador para que espere de forma asíncrona mientras envía otras solicitudes o espera a varios recursos de forma simultánea.

Para obtener acceso a una promesa de un esperador, recupérela de un cliente utilizando el método `getWaiter($name, array $args = [])` del cliente. Utilice el método `promise()` de un esperador

para iniciarlo. Una promesa de esperador se cumple con la última `Aws\CommandInterface` ejecutada en el esperador y rechazada con una excepción `RuntimeException` al producirse un error.

```
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();
```

Exponer la API de esperadores basados en promesas permite casos de uso potentes y con relativamente poca sobrecarga. Por ejemplo, ¿qué sucede si desea esperar a varios recursos y hacer algo con el primer esperador que se resuelva correctamente?

```
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();
```

Expresiones de JMESPath en el AWS SDK for PHP versión 3

[JMESPath](#) le permite especificar de forma declarativa cómo extraer elementos de un documento JSON. El AWS SDK for PHP depende de [jmespath.php](#) para sustentar algunas de sus abstracciones de alto nivel, como los [paginadores del AWS SDK para PHP versión 3](#) (p. 82) y los [esperadores del AWS SDK para](#)

[PHP versión 3 \(p. 84\)](#), pero también expone la búsqueda de JMESPath en `Aws\ResultInterface` y `Aws\ResultPaginator`.

Puede experimentar con JMESPath en su navegador probando los ejemplos online de [JMESPath](#). Obtenga más información sobre el lenguaje, incluidas las expresiones y funciones disponibles, en la [especificación JMESPath](#).

La [CLI de AWS](#) es compatible con JMESPath. Las expresiones que escribe para la salida de la CLI son completamente compatibles con las expresiones escritas para AWS SDK for PHP.

Extraer datos de los resultados

La interfaz `Aws\ResultInterface` tiene un método `search($expression)` que extrae datos de un modelo de resultados en función de una expresión JMESPath. El uso de expresiones JMESPath para consultar los datos de un objeto de resultado puede ayudar a eliminar el código condicional reutilizable y expresar con mayor concisión los datos que se extraen.

Para demostrar cómo funciona, empezaremos con el valor de salida JSON predeterminado, que describe dos volúmenes Amazon Elastic Block Store (Amazon EBS) asociados a instancias Amazon EC2 independientes.

```
$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"
        }
    }
}
```

En primer lugar, podemos obtener solo el primer volumen de la lista Volumes con el comando siguiente.

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

Ahora, utilizamos la expresión wildcard-index para recorrer en iteración toda la lista y también extraemos y cambiamos el nombre de tres elementos: [*] El nombre de VolumeId se cambia a ID, el nombre de AvailabilityZone se cambia a AZ y el de Size sigue siendo Size. Podemos extraer y cambiar el nombre de estos elementos mediante una expresión multi-hash situada después de la expresión wildcard-index.

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

Esto resulta en una matriz de datos PHP como la siguiente:

```
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)
    }
}
```

En la notación multi-hash también puede utilizar claves encadenadas como key1.key2[0].key3 para extraer los elementos profundamente anidados en la estructura. El siguiente ejemplo ilustra esto con la clave Attachments[0].InstanceId, con el simple alias InstanceId. : (en la mayoría de los casos, las expresiones JMESPath no tendrán en cuenta los espacios en blanco).

```
$expr = 'Volumes[*].{ID: VolumeId,
                    InstanceId: Attachments[0].InstanceId,
```

```
AZ: AvailabilityZone,  
Size: Size}';  
  
$data = $result->search($expr);  
var_dump($data);
```

La expresión anterior devolverá los datos siguientes:

```
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)  
    }  
}
```

También puede filtrar varios elementos con la expresión multi-list: [key1, key2]. Con ella todos los atributos filtrados adoptan la forma de una única lista ordenada por cada objeto, con independencia del tipo.

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

La ejecución de la búsqueda anterior produce los datos siguientes:

```
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)  
}  
}
```

Utilice una expresión `filter` para filtrar los resultados por el valor de un campo específico. El siguiente ejemplo de consulta devuelve únicamente los volúmenes de la zona de disponibilidad `us-west-2a`.

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

también admite expresiones de función. JMESPath Supongamos que desea ejecutar la misma consulta que en el ejemplo anterior, pero obteniendo todos los volúmenes de las regiones de AWS que comiencen por "us-". La expresión siguiente utiliza la función `starts_with` y pasa una cadena literal de `us-`. El resultado de esta función se compara entonces con el valor literal JSON `true`, pasando solo los resultados del predicado de filtro que hayan devuelto `true` en la proyección del filtro.

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

Extraer datos de paginadores

Como se indicó en [Paginadores en el AWS SDK para PHP versión 3 \(p. 82\)](#), los objetos `Aws\ResultPaginator` se usan para obtener los resultados de una operación paginable de la API. El AWS SDK for PHP le permite extraer e iterar por datos filtrados desde objetos `Aws\ResultPaginator`, básicamente mediante la implementación de un [mapa plano](#) sobre el iterador en el que el resultado de una expresión JMESPath es la función de mapa.

Supongamos que desea crear un objeto `iterator` que solo devuelva los objetos de un bucket con un tamaño superior a 1 MB. Esto puede conseguirse creando primero un paginador `ListObjects` y aplicándole entonces una función `search()` para crear un iterador de mapa plano para los datos paginados.

```
$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);  
}
```

Métricas de SDK en la AWS SDK for PHP versión 3

AWS SDK Metrics for Enterprise Support (Métricas de SDK) permite a los clientes empresariales recopilar métricas de AWS SDKs en sus hosts y clientes compartidos con AWS Enterprise Support. SDK Metrics proporciona información que ayuda a acelerar la detección y el diagnóstico de los problemas que se producen en las conexiones con los servicios de AWS y que afectan a los clientes de AWS Enterprise Support.

A medida que la telemetría se recopila en cada host, se transmite a través de UDP a localhost, donde el agente de CloudWatch agrega los datos y los envía al servicio de Métricas de SDK. Por lo tanto, para recibir métricas, debe añadir el cliente de CloudWatch a su instancia.

Encontrará más información acerca de [SDK Metrics](#) en la Amazon CloudWatch User Guide.

En cada uno de los temas siguientes se describe cómo configurar y administrar Métricas de SDK para el AWS SDK for PHP.

Temas

- [Describe cómo autorizar a Métricas de SDK para que recopile y envíe métricas en el AWS SDK for PHP versión 3 \(p. 91\)](#)
- [Configuración de Métricas de SDK para el AWS SDK for PHP versión 3 \(p. 93\)](#)
- [Definiciones de Métricas de SDK \(p. 97\)](#)

Describe cómo autorizar a Métricas de SDK para que recopile y envíe métricas en el AWS SDK for PHP versión 3

Para recopilar métricas de AWS SDKs mediante clientes de AWS SDK Metrics for Enterprise Support Enterprise, debe crear un rol de IAM que conceda permiso al agente de CloudWatch para recopilar datos de su instancia o entorno de producción de Amazon EC2.

Utilice el siguiente ejemplo de código PHP o la consola de AWS para crear una política y un rol de IAM para que un agente de CloudWatch tenga acceso a Métricas de SDK en su entorno.

Obtenga más información sobre el uso de Métricas de SDK con el AWS SDK for PHP en [Configuración de SDK Metrics para el AWS SDK para PHP Version 3 \(p. 93\)](#). Para obtener más información sobre Métricas de SDK, consulte [Definición de permisos de IAM para SDK Metrics](#) en la Amazon CloudWatch User Guide.

Configurar los permisos de acceso mediante el AWS SDK for PHP

Debe crear un rol de IAM para la instancia que tenga permiso para Amazon EC2 Systems Manager y Métricas de SDK.

En primer lugar, cree una política con [CreatePolicy](#). A continuación, cree un rol con [CreateRole](#). Por último, asocie la política que ha creado al nuevo rol con [AttachRolePolicy](#).

Importaciones

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

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

Código de muestra

```
$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());  
}
```

Configuración de los permisos de acceso mediante la consola de IAM

También puede utilizar la consola de IAM para crear un rol.

1. Vaya a la [consola de IAM](#) y cree un rol para usar Amazon EC2.
2. Seleccione Roles en el panel de navegación.
3. Elija Create Role.
4. Elija AWS Service (Servicio de AWS) y, a continuación, EC2.
5. Seleccione Next: (Siguiente:). Permisos.
6. En Attach permissions policies (Asociar políticas de permisos), elija create policy (crear política).
7. En Service (Servicio), elija Systems Manager. En Actions (Acciones), expanda Read y elija GetParameters. Para los recursos, especifique su agente de CloudWatch.
8. Añada un permiso adicional.
9. Seleccione Choose a service (Elija un servicio) y, a continuación, Enter service manually (Especificar el servicio manualmente). En Service (Servicio), escriba `sdkmetrics`. Seleccione todas las acciones de `sdkmetrics` y todos los recursos y, a continuación, elija Review Policy (Revisar política).
10. Asigne un nombre al rol `AmazonSDKMetrics` y añada una descripción.
11. Elija Create Role.

Configuración de Métricas de SDK para el AWS SDK for PHP versión 3

En los siguientes pasos, se muestra cómo configurar Métricas de SDK para el AWS SDK for PHP. Estos pasos corresponden a una instancia Amazon EC2 que ejecuta Amazon Linux para una aplicación cliente que utiliza el AWS SDK for PHP. Métricas de SDK también está disponible para entornos de producción si se habilita durante la configuración del AWS SDK for PHP.

Para usar Métricas de SDK, ejecute la última versión del agente de CloudWatch. Obtenga información sobre cómo [configurar el agente de CloudWatch para métricas de SDK](#) en la Amazon CloudWatch User Guide.

Para obtener más información sobre los permisos de IAM para Métricas de SDK, consulte el tema sobre [permisos de IAM para SDK Metrics para el AWS SDK para PHP \(p. 91\)](#).

Para configurar Métricas de SDK con el AWS SDK for PHP, siga estas instrucciones:

1. Cree una aplicación con un cliente de AWS SDK for PHP para utilizar un servicio de AWS.
2. Aloje el proyecto en una instancia Amazon EC2 o en su entorno local.

3. Instale y utilice la versión más reciente del AWS SDK for PHP.
4. Instale y configure un agente de CloudWatch en una instancia EC2 o en su entorno local.
5. Autorice a Métricas de SDK para que recopile y envíe métricas.
6. [Habilite SDK Metrics para el AWS SDK para PHP \(p. 94\)](#).

Para obtener más información, consulte los siguientes temas:

- [Actualización de un agente de CloudWatch \(p. 96\)](#)
- [Deshabilitación de SDK Metrics \(p. 97\)](#)

Habilitar Métricas de SDK para el AWS SDK for PHP

De forma predeterminada, Métricas de SDK está desactivado, el host está configurado en "127.0.0.1" y el puerto está establecido en 31000. Los siguientes son los parámetros predeterminados.

```
//default values
[
    'enabled' => false,
    'host' => '127.0.0.1',
    'port' => 31000,
]
```

La habilitación de Métricas de SDK es independiente de la configuración de sus credenciales para utilizar un servicio de AWS.

Puede habilitar Métricas de SDK si pasa una opción de configuración de cliente, configura variables de entorno o utiliza el archivo de configuración compartida de AWS.

Orden de prioridad

El orden de prioridad es el siguiente (1 sobrescribe 2-3, etc.):

1. Opción de configuración de cliente
2. Variables de entorno
3. Archivo de configuración compartida de AWS

Opción 1: Opción de configuración del cliente

Puede pasar una opción `csm` a su constructor cliente para habilitar y establecer las opciones de configuración. Esta opción puede ser una matriz asociativa, una instancia de `\Aws\ClientIdSideMonitoring\ConfigurationInterface`, un invocable que proporciona una instancia de `\Aws\ClientIdSideMonitoring\ConfigurationInterface` o el valor booleano de `false`.

Matriz asociativa:

```
$client = new \Aws\S3\S3Client([
    'region' => 'your-region',
    'version' => 'latest',
    'csm' => [
        'enabled' => true,
        'host' => 'my.host',
        'port' => 1234,
        'client_id' => 'My Application'
    ]
]);
```

Instancia de `\Aws\ClientSideMonitoring\ConfigurationInterface`:

```
$client = new \Aws\S3\S3Client([
    'region' => 'your-region',
    'version' => 'latest',
    'csm' => new \Aws\ClientSideMonitoring\Configuration(
        true,
        'my.host',
        1234,
        'My Application'
    )
]);
```

Invocable:

```
$client = new \Aws\S3\S3Client([
    'region' => 'your-region',
    'version' => 'latest',
    'csm' => function() {
        return new \Aws\ClientSideMonitoring\Configuration(
            true,
            '127.0.0.1',
            1234,
            'My Application'
        );
    }
]);
```

"Falso" booleano:

```
$client = new \Aws\S3\S3Client([
    'region' => 'your-region',
    'version' => 'latest',
    'csm' => false
]);
```

Opción 2: Configuración de las variables de entorno

El SDK primero verifica el perfil especificado en la variable de entorno en `AWS_PROFILE` para determinar si Métricas de SDK está habilitado.

Para activar Métricas de SDK, añada lo siguiente a las variables de entorno.

```
export AWS_CSM_ENABLED=true
```

Hay [otras opciones de configuración \(p. 96\)](#) disponibles. Para obtener más información acerca de cómo utilizar archivos compartidos, consulte [Uso de credenciales desde las variables de entorno \(p. 46\)](#).

Note

La habilitación de Métricas de SDK no configura sus credenciales para utilizar un servicio de AWS. Para hacerlo, consulte [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#).

Opción 3: Archivo de configuración compartida de AWS

Si no se encuentra ninguna configuración de CSM en las variables de entorno, el SDK busca el campo de perfil de AWS predeterminado en un archivo de configuración compartida. Normalmente, este se encuentra en `~/.aws/config`, pero puede cambiar el archivo que se utiliza estableciendo la variable de entorno `AWS_CONFIG_FILE`. Si `AWS_DEFAULT_PROFILE` se configura de forma distinta a la predeterminada,

actualice ese perfil. Para habilitar SDK Metrics, agregue `csm_enabled` al archivo de configuración compartida.

```
[default]
csm_enabled = true

[profile aws_csm]
csm_enabled = true
```

Hay [otras opciones de configuración \(p. 96\)](#) disponibles. Para obtener más información acerca de cómo utilizar los archivos compartidos de AWS, consulte [Uso del archivo de credenciales de AWS y perfiles de credenciales \(p. 46\)](#).

Note

La habilitación de Métricas de SDK no configura sus credenciales para utilizar un servicio de AWS. Para hacerlo, consulte [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#).

Actualización de un agente de CloudWatch

Para realizar cambios en el host o el ID del puerto, debe establecer los valores y, a continuación, reiniciar los trabajos AWS que estén activos en ese momento.

Opción 1: Opción de configuración del cliente

Consulte la opción 1 más arriba en "Habilitar Métricas de SDK para AWS SDK for PHP" para ver ejemplos de cómo establecer la opción `csm` de configuración del cliente para modificar la configuración de CSM.

Opción 2: Configuración de las variables de entorno

La mayoría de los servicios de AWS utilizan el puerto predeterminado. Pero si el servicio que desea que monitoree Métricas de SDK utiliza un puerto distinto, añada `AWS_CSM_PORT=[número_de_puerto]` a las variables de entorno del host. Además, se puede especificar un host distinto mediante la variable de entorno `AWS_CSM_HOST`.

```
export AWS_CSM_ENABLED=true
export AWS_CSM_PORT=1234
export AWS_CSM_HOST=192.168.0.1
```

Opción 3: Archivo de configuración compartida de AWS

La mayoría de los servicios utilizan el puerto predeterminado. Pero si su servicio requiere un ID de puerto único, agregue `csm_port = [port_number]` a `~/.aws/config` (o el archivo especificado por la variable de entorno `AWS_CONFIG_FILE`). Un host no predeterminado se puede configurar mediante `csm_host`.

```
[default]
csm_enabled = false
csm_host = 123.4.5.6
csm_port = 1234

[profile aws_csm]
csm_enabled = false
csm_host = 123.4.5.6
csm_port = 1234
```

Reinicio de Métricas de SDK

Para reiniciar un trabajo, ejecute los siguientes comandos.

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

Deshabilitar Métricas de SDK

Para desactivar Métricas de SDK, establezca `csm_enabled` en `false` en las variables de entorno o en el archivo de configuración compartida de AWS ubicado en `~/.aws/config` (o en el archivo especificado por la variable de entorno `AWS_CONFIG_FILE`). A continuación, reinicie su agente de CloudWatch para que los cambios puedan surtir efecto.

Establezca `csm_enabled` en `false`

Note

Tenga en cuenta el orden de prioridad indicado anteriormente. Por ejemplo, si se ha habilitado Métricas de SDK en las variables de entorno pero se ha deshabilitado en el archivo de configuración, Métricas de SDK permanece habilitado.

Opción 1: Configuración de cliente

```
$client = new \Aws\S3\S3Client([  
    'region' => 'your-region',  
    'version' => 'latest',  
    'csm' => false  
]);
```

Opción 2: Variables de entorno

```
export AWS_CSM_ENABLED=false
```

Opción 3: Archivo de configuración compartida de AWS

```
[default]  
csm_enabled = false  
  
[profile aws_csm]  
csm_enabled = false
```

Detención de Métricas de SDK y reinicio del agente de CloudWatch

Para deshabilitar Métricas de SDK, utilice el siguiente comando.

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

Si utiliza otras características de CloudWatch, reinicie CloudWatch con el comando siguiente.

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

Definiciones de Métricas de SDK

Puede utilizar las siguientes descripciones de Métricas de SDK para interpretar los resultados. En general, estas métricas están disponibles para que las revise con el director de cuentas técnicas durante las

revisiones de negocio periódicas. Los recursos de AWS Support y su director de cuentas técnicas deben tener acceso a los datos de SDK Metrics para ayudarle a resolver los casos, pero si encuentra datos que son confusos o inesperados, aunque no parezca que afectan negativamente al rendimiento de sus aplicaciones, es mejor revisar esos datos durante las revisiones de negocio programadas.

Métrica:	CallCount
Definición	Número total de llamadas a la API realizadas correctamente o con errores desde el código a los servicios de AWS.
Modo de uso	Utilícela como línea de referencia para correlacionarla con otras métricas como errores o limitación controlada.

Métrica:	ClientErrorCount
Definición	Número de llamadas a la API que se realizaron incorrectamente con errores del cliente (códigos de respuesta HTTP 4xx). Ejemplos: Limitación controlada, acceso denegado, el bucket de S3 no existe y valor de parámetro no válido.
Modo de uso	Excepto en ciertos casos relacionados con la limitación controlada (por ejemplo, cuando se produce debido a un límite que se debe incrementar) esta métrica puede indicar algo en su aplicación que se debe solucionar.

Métrica:	ConnectionErrorCount
Definición	Número de llamadas a la API que no se realizan correctamente debido a errores de conexión con el servicio. Estos errores pueden deberse a problemas de red entre la aplicación del cliente y los servicios de AWS, incluidos los balanceadores de carga, errores de DNS y proveedores de tránsito. En algunos casos, los problemas de AWS pueden dar lugar a este error.
Modo de uso	Utilice esta métrica para determinar si los problemas son específicos de la aplicación o los causa la infraestructura o la red. Un ConnectionErrorCount elevado también podría indicar valores de tiempo de espera cortos para las llamadas a la API.

Métrica:	ThrottleCount
Definición	Número de llamadas a la API que producen errores debido a la limitación controlada impuesta por los servicios de AWS.

Métrica:	ThrottleCount
Modo de uso	Utilice esta métrica para evaluar si la aplicación ha alcanzado los límites, así como para determinar la causa de los reintentos y la latencia de la aplicación. Considere la posibilidad de distribuir las llamadas a través de una ventana en lugar de agruparlas por lotes.

Métrica:	ServerErrorCount
Definición	Número de llamadas a la API que se realizan incorrectamente debido a errores de servidor (códigos de respuesta HTTP 5xx) desde AWS Services. Suelen ser causados por los servicios de AWS.
Modo de uso	Determine la causa de los reintentos o latencia del SDK. Esta métrica no siempre indica que los servicios de AWS son los causantes de los errores, ya que algunos equipos de AWS clasifican la latencia como una respuesta HTTP 503.

Métrica:	EndToEndLatency
Definición	Tiempo total del que dispone la aplicación para realizar una llamada utilizando el AWS SDK, incluidos los reintentos. En otras palabras, independientemente de si se realiza correctamente después de varios intentos, o tan pronto como no se produce una llamada debido a un error irreparable.
Modo de uso	Determine cómo contribuyen las llamadas a la API de AWS a la latencia general de la aplicación. Una latencia superior a la esperada puede deberse a problemas con la red, el firewall u otros parámetros de configuración o a la latencia que se produce como resultado de los reintentos del SDK.

Uso de los servicios de AWS con el AWS SDK for PHP versión 3

Algunos de los servicios de AWS que son compatibles con el AWS SDK for PHP versión 3 incluyen características adicionales además de ejecutar operaciones en una API. Esta guía documenta estas características de nivel superior para cada servicio.

Temas

- [Uso de AWS Cloud9 con el AWS SDK for PHP versión 3 \(p. 100\)](#)
- [Uso del administrador de sesiones de DynamoDB con el AWS SDK for PHP versión 3 \(p. 102\)](#)
- [Cliente de Amazon S3 para varias regiones con el AWS SDK for PHP versión 3 \(p. 107\)](#)
- [Encapsulador de flujo de Amazon S3 con el AWS SDK for PHP versión 3 \(p. 108\)](#)
- [Gestor de transferencias de Amazon S3 para AWS SDK for PHP con el versión 3 \(p. 113\)](#)
- [Cifrado del lado del cliente de Amazon S3 con el AWS SDK for PHP versión 3 \(p. 116\)](#)

Uso de AWS Cloud9 con el AWS SDK for PHP versión 3

Puede utilizar AWS Cloud9 con AWS SDK for PHP versión 3 para escribir, ejecutar y depurar el código PHP con tan solo un navegador. AWS Cloud9 incluye herramientas, como un editor de código, un depurador y un terminal. Dado que el IDE de AWS Cloud9 está basado en la nube, puede trabajar en sus proyectos desde la oficina, desde su casa o desde cualquier lugar con un equipo conectado a Internet. Para obtener más información general sobre AWS Cloud9, consulte la [Guía de usuario de AWS Cloud9](#).

Siga estas instrucciones para configurar AWS Cloud9 con AWS SDK for PHP:

- [Paso 1: Configuración de su cuenta de AWS para utilizar AWS Cloud9 \(p. 100\)](#)
- [Paso 2: Configuración del entorno de desarrollo de AWS Cloud9 \(p. 101\)](#)
- [Paso 3: Configuración de AWS SDK para PHP \(p. 101\)](#)
- [Paso 4: Descargar código de ejemplo \(p. 101\)](#)
- [Paso 5: Ejecución y depuración del código de ejemplo \(p. 102\)](#)

Paso 1: Configurar su cuenta de AWS para usar AWS Cloud9

Para empezar a utilizar AWS Cloud9, inicie sesión en la consola de AWS Cloud9 como entidad de AWS Identity and Access Management (IAM) (por ejemplo, un usuario de IAM) en su cuenta de AWS que tiene permisos de acceso para AWS Cloud9.

Para configurar una entidad de IAM en su cuenta de AWS para acceder a AWS Cloud9 e iniciar sesión en la consola de AWS Cloud9, consulte la sección sobre [configuración de equipos para AWS Cloud9](#) en la [Guía del usuario de AWS Cloud9](#).

Paso 2: Configuración del entorno de desarrollo de AWS Cloud9

Después de iniciar sesión en la consola de AWS Cloud9, utilice la consola para crear un entorno de desarrollo de AWS Cloud9. Una vez creado el entorno, AWS Cloud9 abrirá el IDE en dicho entorno.

Consulte la sección sobre [creación de un entorno en AWS Cloud9](#) en la [Guía de usuario de AWS Cloud9](#) para obtener más información.

Note

Al crear el entorno en la consola por primera vez, le recomendamos que elija la opción **Create a new instance for environment (EC2)** [Crear una nueva instancia para el entorno (EC2)]. Esta opción le indica a AWS Cloud9 que cree un entorno, lance una instancia Amazon EC2 y, a continuación, conecte la nueva instancia al nuevo entorno. Esta es la forma más rápida de empezar a utilizar AWS Cloud9.

Paso 3: Configuración de la AWS SDK for PHP de

Después de que AWS Cloud9 abra el IDE para su entorno de desarrollo, utilice el IDE para configurar AWS SDK for PHP en su entorno, tal y como se indica a continuación.

1. Si el terminal todavía no está abierto en el IDE, ábralo. En la barra de menú del IDE, elija **Window, New Terminal** (Ventana, Nuevo terminal).
2. Ejecute los siguientes comandos para instalar AWS SDK for PHP.

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

Si el IDE no encuentra el PHP, ejecute el siguiente comando para instalarlo. (Este comando presupone que eligió la opción de **Create a new instance for environment (EC2)** [Crear una nueva instancia para el entorno (EC2)], vista anteriormente en este tema).

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

Paso 4: Descargar código de ejemplo

Utilice el terminal que ha abierto en el paso anterior para descargar un código de ejemplo para el AWS SDK for PHP en el entorno de desarrollo de AWS Cloud9.

Para ello, ejecute el siguiente comando. Este comando descarga una copia de todos los ejemplos de código que se utilizan en la documentación oficial del SDK de AWS en el directorio raíz del entorno.

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

Para encontrar ejemplos de código para AWS SDK for PHP, utilice la ventana **Environment** (Entorno) para abrir el directorio `ENVIRONMENT_NAMEaws-doc-sdk-examplesphpexample_code`, donde `ENVIRONMENT_NAME` es el nombre de su entorno de desarrollo.

Para obtener información sobre cómo trabajar con estos y otros ejemplos de código, consulte la sección sobre [ejemplos de código](#) (p. 122).

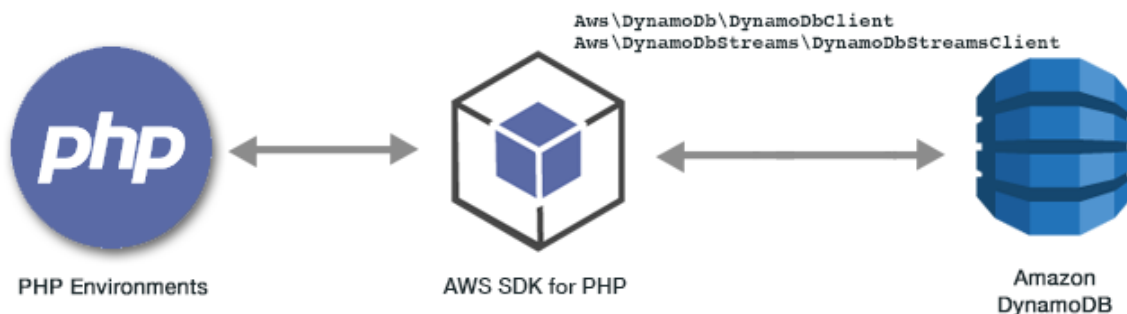
Paso 5: Ejecución y depuración del código de ejemplo

Para ejecutar código en su entorno de desarrollo de AWS Cloud9, consulte la sección sobre [cómo ejecutar su código](#) en la [Guía del usuario de AWS Cloud9](#).

Para depurar código, consulte la sección sobre cómo [depurar su código](#) en la [Guía del usuario de AWS Cloud9](#).

Uso del administrador de sesiones de DynamoDB con el AWS SDK for PHP versión 3

El administrador de sesiones de DynamoDB es un administrador de sesiones personalizado para PHP que permite a los desarrolladores utilizar Amazon DynamoDB como almacén de sesiones. Si utiliza DynamoDB para el almacenamiento de sesiones evitará que surjan problemas al gestionar sesiones en una aplicación web distribuida al trasladar las sesiones fuera del sistema de archivos local a una ubicación compartida. DynamoDB es rápido, escalable, fácil de configurar y administra la replicación de los datos de forma automática.



El administrador de sesiones de DynamoDB utiliza la función `session_set_save_handler()` para enlazar operaciones de DynamoDB con [funciones de sesión nativas](#) de PHP para permitir una verdadera caída en su lugar. Esto incluye admitir características como el bloqueo de sesiones y la recopilación de elementos no utilizados, que forman parte del administrador de sesiones predeterminado de PHP.

Para obtener más información sobre el servicio de DynamoDB, consulte la [página de inicio de Amazon DynamoDB](#).

Uso básico

Paso 1: Registrar el controlador

En primer lugar, cree instancias y registre el administrador de sesiones.

```
use Aws\DynamoDb\SessionHandler;

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

$sessionHandler->register();
```

Paso 2. Crear una tabla para almacenar sus sesiones

Antes de poder empezar a usar el administrador de sesiones, debe crear una tabla en la que se almacenen las sesiones. Puede hacerlo previamente utilizando la [consola de AWS para Amazon DynamoDB](#) o la AWS SDK for PHP.

Al crear esta tabla utilice "id" como nombre de la clave principal. También se recomienda configurar un [atributo Tiempo de vida](#) utilizando el atributo "expires" para beneficiarse de la recopilación automática de elementos no utilizados de las sesiones.

Paso 3. Utilizar las sesiones de PHP como suele hacerlo

Una vez registrado el administrador de sesiones y cuando la tabla ya exista, podrá escribir y leer desde la sesión utilizando la `$_SESSION` superglobal, tal como suele hacer con el administrador de sesiones predeterminado de PHP. El administrador de sesiones de DynamoDB encapsula y abstrae las interacciones con DynamoDB y le permite utilizar las funciones de sesión nativas y la interfaz de PHP.

```
// Start the session
session_start();

// Alter the session data
$_SESSION['user.name'] = 'jeremy';
$_SESSION['user.role'] = 'admin';

// Close the session (optional, but recommended)
session_write_close();
```

Configuration

Puede configurar el comportamiento del administrador de sesiones utilizando las siguientes opciones. Todas las opciones son opcionales, pero asegúrese de que comprende cuáles son los valores predeterminados.

table_name

Es el nombre de la tabla de DynamoDB en la que se almacenan las sesiones. El valor predeterminado es 'sessions'.

hash_key

Es el nombre de la clave hash en la tabla de sesiones de DynamoDB. El valor predeterminado es 'id'.

data_attribute

El nombre del atributo en la tabla de sesiones DynamoDB en la que se almacenan los datos de la sesión. El valor predeterminado es 'data'.

data_attribute_type

El tipo del atributo de la tabla de sesiones DynamoDB en la que se almacenan los datos de la sesión. El valor predeterminado es 'string', pero se puede establecer de forma opcional en 'binary'.

session_lifetime

Es la vida útil de una sesión inactiva antes de que se recopile como elemento no utilizado. Si no es el caso, el valor de la vida útil real que se utilizará es `ini_get('session.gc_maxlifetime')`.

session_lifetime_attribute

El nombre del atributo en la tabla de sesiones DynamoDB en la que se almacena la hora de vencimiento de la sesión. El valor predeterminado es 'expires'.

consistent_read

Indica si el administrador de sesiones debería utilizar lecturas consistentes para la operación `GetItem`. El valor predeterminado es `true`.

locking

Indica si se utiliza el bloqueo de sesiones. El valor predeterminado es `false`.

batch_config

Es la configuración utilizada para la eliminación de lotes al recopilar elementos no utilizados. Estas opciones se transfieren directamente a objetos [DynamoDB WriteRequestBatch](#). Active manualmente la recopilación de elementos no utilizados mediante `SessionHandler::garbageCollect()`.

max_lock_wait_time

Es el tiempo máximo (en segundos) que debe esperar el administrador de sesiones para adquirir un bloqueo antes de desistir. El valor predeterminado es 10 y solo se utiliza con el bloqueo de sesiones.

min_lock_retry_microtime

Es el tiempo mínimo (en microsegundos) que debe esperar el administrador de sesiones entre intentos para adquirir un bloqueo. El valor predeterminado es 10000 y solo se utiliza con el bloqueo de sesiones.

max_lock_retry_microtime

Es el tiempo máximo (en microsegundos) que debe esperar el administrador de sesiones entre intentos para adquirir un bloqueo. El valor predeterminado es 50000 y solo se utiliza con el bloqueo de sesiones.

Para configurar el administrador de sesiones, especifique las opciones de configuración cuando cree instancias del administrador. El siguiente código es un ejemplo de todas las opciones de configuración.

```
$sessionHandler = SessionHandler::fromClient($dynamoDb, [  
    'table_name'           => 'sessions',  
    'hash_key'            => 'id',  
    'data_attribute'       => 'data',  
    'data_attribute_type'  => 'string',  
    'session_lifetime'     => 3600,  
    'session_lifetime_attribute' => 'expires',  
    'consistent_read'      => true,  
    'locking'              => false,  
    'batch_config'         => [],  
    'max_lock_wait_time'   => 10,  
    'min_lock_retry_microtime' => 5000,  
    'max_lock_retry_microtime' => 50000,  
]);
```

Pricing

Aparte del almacenamiento de datos y las tarifas de transferencia de datos, los costos asociados al uso de DynamoDB se calculan en función de la capacidad de desempeño provisionada de la tabla (consulte la [información sobre precios de Amazon DynamoDB](#)). El rendimiento se mide en unidades de capacidad de escritura y capacidad de lectura. La página de inicio de Amazon DynamoDB dice lo siguiente:

Una unidad de capacidad de lectura representa una lectura de consistencia alta por segundo (o bien dos lecturas consistentes finales por segundo) para elementos de un tamaño de hasta 4 KB. Una unidad de capacidad de escritura representa una escritura por segundo para elementos de un tamaño de hasta 1 KB.

En definitiva, el rendimiento y los costos necesarios para su tabla de sesiones se corresponden con el tráfico esperado y el tamaño de la sesión. En la tabla siguiente se explica la cantidad de operaciones de lectura y de escritura que se realiza en su tabla de DynamoDB para cada una de las funciones de la sesión.

Leer mediante <code>session_start()</code>	• 1 operación de lectura (solo 0,5 si <code>consistent_read</code> es <code>false</code>). • (Condicional) 1 operación de escritura para eliminar la sesión si ha caducado.
Leer mediante <code>session_start()</code> (utilizando el bloqueo de sesiones)	• 1 operación de escritura como mínimo. • (Condicional) Las operaciones de escritura adicionales para cada intento de adquisición de un bloqueo en la sesión. En función del tiempo de espera del bloqueo configurado y de las opciones de reintento. • (Condicional) 1 operación de escritura para eliminar la sesión si ha caducado.
Escribir mediante <code>session_write_close()</code>	• 1 operación de escritura.
Eliminar mediante <code>session_destroy()</code>	• 1 operación de escritura.
Recopilación de elementos no utilizados	• 0,5 operaciones de lectura por 4 KB de datos de la tabla para buscar sesiones caducadas. • 1 operación de escritura por elemento caducado para eliminarlo.

Bloqueo de sesiones

El administrador de sesiones de DynamoDB es compatible con el bloqueo de sesiones pesimista para imitar el comportamiento del administrador de sesiones predeterminado de PHP. De forma predeterminada, el administrador de sesiones de DynamoDB tiene esta característica desactivada, ya que puede convertirse en un cuello de botella de rendimiento e incrementar los costos, especialmente cuando una aplicación accede a la sesión utilizando solicitudes Ajas o iframes. Es importante plantearse si la aplicación requiere el bloqueo de sesiones antes de habilitarlo.

Para habilitar el bloqueo de sesiones, establezca la opción `'locking'` en `true` cuando cree instancias de `SessionHandler`.

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

Recopilación de elementos no utilizados

Configure un atributo TTL en la tabla de DynamoDB utilizando el atributo `"expires"`. De este modo, se realizará automáticamente la recopilación de elementos no utilizados de las sesiones y se evitará la necesidad de realizarla usted mismo.

El administrador de sesiones de DynamoDB también admite la recopilación de elementos no utilizados de las sesiones mediante el uso de una serie de operaciones `Scan` y `BatchWriteItem`. Debido a la

naturaleza del funcionamiento de la operación `Scan`, y para encontrar todas las sesiones caducadas y eliminarlas, el proceso de recopilación de elementos no utilizados puede requerir una gran cantidad de rendimiento provisionado.

Por este motivo, no se admite la recopilación automatizada. Una de las prácticas recomendadas es programar la recopilación fuera de las horas punta, cuando la ráfaga de rendimiento consumido no afecte al resto de la aplicación. Por ejemplo, puede configurar un trabajo cron por la noche que dispare un script para ejecutar la recopilación. Este script debería hacer algo similar a lo siguiente.

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

$sessionHandler->garbageCollect();
```

También puede utilizar la opción `'before'` dentro de `'batch_config'` para introducir retrasos en las operaciones `BatchWriteItem` que lleva a cabo el proceso de recopilación de elementos no utilizados. Esto aumentará el tiempo que se tarda en completar la recopilación, pero puede ayudarle a distribuir las solicitudes realizadas por el administrador de sesiones de DynamoDB para mantenerse cerca o dentro de su capacidad de rendimiento provisionada durante la recopilación.

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

$sessionHandler->garbageCollect();
```

Prácticas recomendadas de

1. Cree su tabla de sesiones en una región geográfica de AWS más cercana o en la misma región donde se encuentran los servidores de aplicaciones. De este modo, se consigue la latencia más baja entre su aplicación y la base de datos de DynamoDB.
2. Elija la capacidad de rendimiento provisionada de su tabla de sesiones detenidamente. Tenga en cuenta el tráfico esperado en su aplicación y el tamaño previsto de sus sesiones. También puede utilizar el modo de capacidad de lectura/escritura "bajo demanda" para la tabla.
3. Monitoree su rendimiento utilizado en la consola de administración de AWS o con Amazon CloudWatch y ajuste la configuración de su rendimiento según sea necesario para satisfacer las exigencias de su aplicación.
4. Intente mantener un tamaño reducido de sus sesiones (idealmente de menos de 1 KB). Las sesiones pequeñas funcionan mejor y requieren menos capacidad de rendimiento provisionada.
5. No utilice el bloqueo de sesiones a menos que lo requiera su aplicación.
6. En lugar de utilizar disparadores de la recopilación de elementos no utilizados en las sesiones integrados en PHP, programe su recopilación mediante un trabajo cron u otro mecanismo de programación, para que se ejecute en las horas de menor actividad. Aproveche la opción `'batch_config'`.

Permisos de IAM necesarios

Para utilizar la DynamoDB SessionHandler, sus [credenciales configuradas \(p. 44\)](#) deben tener permiso para utilizar la tabla DynamoDB que [creó en un paso anterior \(p. 103\)](#). La política de IAM siguiente contiene los permisos mínimos necesarios. Para utilizar esta política, sustituya el valor Resource por el Amazon Resource Name (ARN) de la tabla que creó anteriormente. Para obtener más información sobre cómo crear y asociar políticas de IAM, consulte [Administración de políticas de IAM](#) en la IAM User Guide.

```
{
  "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>"
    }
  ]
}
```

Cliente de Amazon S3 para varias regiones con el AWS SDK for PHP versión 3

El AWS SDK for PHP versión 3 proporciona un cliente genérico para varias regiones que se puede utilizar con cualquier servicio. Esto permite a los usuarios especificar la región de AWS a la que quieren enviar un comando proporcionando un parámetro de entrada `@region` a cualquier comando. Además, el SDK proporciona un cliente de Amazon S3 para varias regiones que responde de manera inteligente a determinados errores de Amazon S3 y que redirige los comandos en consecuencia. Esto permite a los usuarios utilizar el mismo cliente para comunicarse con varias regiones. Se trata de una característica especialmente útil para los usuarios del [Encapsulador de flujo de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 108\)](#), cuyos buckets residen en varias regiones.

Uso básico

El patrón de uso básico de un cliente de Amazon S3 es el mismo tanto si se usa un cliente de S3 estándar como su equivalente para varias regiones. La única diferencia de uso en el nivel de comandos es que una región de AWS se puede especificar con el parámetro de entrada `@region`.

```
// 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

Si utiliza el cliente de Amazon S3 para varias regiones, no habrá ninguna excepción de redireccionamiento permanente. Si se envía un comando a la región incorrecta, un cliente de Amazon S3 estándar lanzará una instancia `Aws\S3\Exception\PermanentRedirectException`. En cambio, un cliente para varias regiones reenviará el comando a la región correcta.

Caché de la región del bucket

Los clientes de Amazon S3 para varias regiones mantienen una caché interna de las regiones de AWS en la que residen buckets determinados. De forma predeterminada, cada cliente tiene su propia caché en memoria. Para compartir una memoria caché entre los clientes o procesos, proporcione una instancia de `Aws\CacheInterface` según la opción `bucket_region_cache` para su cliente para varias regiones.

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

Encapsulador de flujo de Amazon S3 con el AWS SDK for PHP versión 3

El encapsulador de flujo de Amazon S3 le permite almacenar y recuperar datos de Amazon S3 utilizando funciones de PHP integradas como, por ejemplo, `file_get_contents`, `fopen`, `copy`, `rename`, `unlink`, `mkdir` y `rmdir`.

Debe registrar el encapsulador flujo de Amazon S3 para utilizarlo.

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

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

De este modo, puede acceder a los buckets y objetos almacenados en Amazon S3 utilizando el protocolo `s3://`. El encapsulador de flujo de Amazon S3 acepta cadenas que contienen un nombre de bucket seguido de una barra inclinada y una clave de objeto opcional o prefijo: `s3://<bucket>[/<key-or-prefix>]:`.

Note

El encapsulador de flujo se ha diseñado para trabajar con objetos y buckets en los que tenga al menos permiso de lectura. Esto significa que su usuario debe tener permiso para ejecutar `ListBucket` en cualquier bucket y `GetObject` en cualquier objeto con el que el usuario necesita interactuar. Por lo tanto, para los casos de uso en los que no tiene este nivel de permiso, le recomendamos que utilice las operaciones de cliente de S3 directamente.

Descargar datos

Puede tomar el contenido de un objeto utilizando `file_get_contents`. Sin embargo, tenga cuidado con esta función, ya que carga todo el contenido del objeto en la memoria.

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

Utilice `fopen()` cuando trabaje con archivos grandes o si necesita transmitir datos desde Amazon S3.

```
// Open a stream in read-only mode
if ($stream = fopen('s3://bucket/key', 'r')) {
    // While the stream is still open
    while (!feof($stream)) {
        // Read 1,024 bytes from the stream
        echo fread($stream, 1024);
    }
    // Be sure to close the stream resource when you're done with it
    fclose($stream);
}
```

Note

Los errores de escritura de archivos solo se devuelven cuando se llama a `fflush`. Esos errores no se devuelven cuando se llama a un `fclose` sin vaciar. El valor devuelto para `fclose` será `true` si cierra el flujo, independientemente de los errores en respuesta a su `fflush` interno. Estos errores tampoco se devuelven al llamar a `file_put_contents` debido a cómo lo implementa PHP.

Abrir flujos rastreables

Los flujos abiertos en modo "r" solo admiten datos que se pueden leer desde el flujo y no se pueden rastrear de forma predeterminada. Esto es así para que los datos se puedan descargar desde Amazon S3 mediante flujos reales, donde los bytes leídos previamente no tengan que almacenarse en la memoria del búfer. Si necesita poder rastrear un flujo, puede transferir `seekable` a las [opciones de contexto del flujo](#) de una función.

```
$context = stream_context_create([
    's3' => ['seekable' => true]
]);

if ($stream = fopen('s3://bucket/key', 'r', false, $context)) {
    // Read bytes from the stream
    fread($stream, 1024);
    // Seek back to the beginning of the stream
    fseek($stream, 0);
    // Read the same bytes that were previously read
    fread($stream, 1024);
    fclose($stream);
}
```

La apertura de flujos rastreables le permite buscar bytes leídos previamente. No se puede pasar directamente a bytes que todavía no se han leído desde el servidor remoto. Para poder volver a llamar los datos leídos previamente, estos se almacenan en el búfer en un flujo de PHP temporal mediante un decorador de flujos. Cuando la cantidad de datos almacenados en la memoria caché supera los 2 MB, los datos del flujo temporal se transfieren de la memoria al disco. Tenga en mente esta posibilidad al descargar archivos grandes desde Amazon S3 utilizando la opción de contexto del flujo `seekable`.

Cargar datos

Puede cargar datos a Amazon S3 mediante `file_put_contents()`.

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

Puede cargar archivos de mayor tamaño realizando un streaming de los datos utilizando `fopen()` y un modo de acceso de flujo "w", "x" o "a". El encapsulador de flujo de Amazon S3 no permite leer y escribir flujos de forma simultánea (por ejemplo, "r+", "w+", etc). Esto se debe a que el protocolo HTTP no permite la lectura y escritura simultáneas.

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

Note

Amazon S3 requiere que se especifique un encabezado de longitud del contenido antes de que se envíe la carga de una solicitud. Por ello, los datos que se cargan en una operación `PutObject` se almacenan internamente en el búfer mediante un flujo temporal de PHP hasta que se vacía o cierra el flujo.

Note

Los errores de escritura de archivos solo se devuelven cuando se llama a `fflush`. Esos errores no se devuelven cuando se llama a un `fclose` sin vaciar. El valor devuelto para `fclose` será `true` si cierra el flujo, independientemente de los errores en respuesta a su `fflush` interno. Estos errores tampoco se devuelven al llamar a `file_put_contents` debido a cómo lo implementa PHP.

Modos fopen

La función `fopen()` de PHP requiere que especifique una opción `$mode`. La opción de modo especifica si se pueden leer o escribir datos en un flujo y si el archivo debe existir al abrir un flujo. El encapsulador de flujo de Amazon S3 admite los siguientes modos.

r	Es un flujo de solo lectura donde el archivo ya debe existir.
w	Es un flujo de solo escritura. Si el archivo ya existe, se sobrescribe.
a	Es un flujo de solo escritura. Si el archivo ya existe, se descarga en un flujo temporal y cualquier escritura en el flujo se agrega a cualquier dato cargado previamente.
x	Es un flujo de solo escritura. Aparece un error si el archivo no existe todavía.

Otras funciones de objeto

Los encapsuladores de flujo permiten que diferentes funciones de PHP integradas trabajen con un sistema personalizado como Amazon S3. Estas son algunas de las funciones que el encapsulador de flujo de Amazon S3 le permite realizar con objetos almacenados en Amazon S3.

unlink()	Elimina un objeto de un bucket. <pre>// Delete an object from a bucket unlink('s3://bucket/key');</pre> Puede transferir cualquiera de las opciones disponibles a la operación DeleteObject para modificar la forma en que se elimina el objeto (por ejemplo, especificando una versión del objeto determinada). <pre>// Delete a specific version of an object from a bucket unlink('s3://bucket/key', stream_context_create(['s3' => ['VersionId' => '123']]));</pre>
filesize()	Obtiene el tamaño de un objeto. <pre>// Get the Content-Length of an object \$size = filesize('s3://bucket/key',);</pre>
is_file()	Comprueba si una URL es un archivo. <pre>if (is_file('s3://bucket/key')) { echo 'It is a file!'; }</pre>
file_exists()	Comprueba si existe un objeto. <pre>if (file_exists('s3://bucket/key')) { echo 'It exists!'; }</pre>
filetype()	Comprueba si una URL se mapea con un archivo o bucket (dir).
file()	Carga el contenido de un objeto en una matriz de líneas. Puede transferir cualquiera de las opciones disponibles a la operación GetObject para modificar la forma en que se descarga el archivo.
filemtime()	Obtiene la última fecha de modificación de un objeto.
rename()	Para cambiar el nombre de un objeto copiándolo y eliminando el original. Puede transferir las opciones disponibles de las operaciones

`CopyObject` y `DeleteObject` a los parámetros de contexto del flujo para modificar la forma en que se copia y elimina el objeto.

Note

Aunque `copy` normalmente trabaja con el encapsulador de flujo de Amazon S3, es posible que algunos errores no se notifiquen correctamente debido a aspectos internos de la función `copy` en PHP. Le recomendamos que utilice una instancia de [AwsS3ObjectCopier](#) en su lugar.

Uso de buckets de

Puede modificar y examinar buckets de Amazon S3 de forma similar a cómo PHP permite la modificación y el recorrido de los directorios en su sistema de archivos.

A continuación se incluye un ejemplo de creación de un bucket.

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

Puede pasar las opciones de contexto del flujo al método `mkdir()` para modificar la forma en que se crea el bucket utilizando los parámetros disponibles para la operación [CreateBucket](#).

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

Puede eliminar buckets utilizando la función `rmdir()`.

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

Note

Los buckets solo se pueden eliminar si están vacíos.

Mostrar el contenido de un bucket

Puede utilizar las funciones de PHP [opendir\(\)](#), [readdir\(\)](#), [rewinddir\(\)](#) y [closedir\(\)](#) en el encapsulador de flujo de Amazon S3 para revisar el contenido de un bucket. Puede transferir los parámetros disponibles para la operación [ListObjects](#) como opciones de contexto de flujo personalizadas a la función de `opendir()` para modificar la forma en que se enumeran los objetos.

```
$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);
}
```

Puede crear una lista recursiva de cada objeto y prefijo en un bucket utilizando el [RecursiveDirectoryIterator](#) de PHP.

```
$dir = 's3://bucket';  
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));  
  
foreach ($iterator as $file) {  
    echo $file->getType() . ': ' . $file . "\n";  
}
```

Otra forma de enumerar el contenido de un bucket de forma recursiva que requiera menos solicitudes HTTP es utilizar la función `Aws\recursive_dir_iterator($path, $context = null)`.

```
<?php  
require 'vendor/autoload.php';  
  
$iter = Aws\recursive_dir_iterator('s3://bucket/key');  
foreach ($iter as $filename) {  
    echo $filename . "\n";  
}
```

Opciones de contexto del flujo

Puede personalizar el cliente utilizado por el encapsulador de flujo o la caché utilizada para grabar en caché la información cargada previamente acerca de buckets y claves, transfiriendo las opciones de contexto del flujo personalizadas.

El encapsulador de flujo es compatible con las siguientes opciones de contexto del flujo en cada operación.

client

Es el objeto `Aws\AwsClientInterface` que se va a utilizar para ejecutar comandos.

cache

Es una instancia de `Aws\CacheInterface` que se va a utilizar para almacenar en caché las estadísticas de archivos obtenidas previamente. De forma predeterminada, el encapsulador de flujo utiliza una caché LRU en memoria.

Gestor de transferencias de Amazon S3 para AWS SDK for PHP con el versión 3

El gestor de transferencias de Amazon S3 para AWS SDK for PHP se utiliza para cargar directorios completos en un bucket de Amazon S3 y descargar buckets completos en un directorio local.

Carga de un directorio local en Amazon S3

El objeto `Aws\S3\Transfer` se utiliza para realizar transferencias. En el siguiente ejemplo se muestra cómo cargar de forma recursiva un directorio local de archivos a un bucket de Amazon S3.

```
// 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();
```

En este ejemplo, hemos creado un cliente de Amazon S3, un objeto `Transfer` y realizado la transferencia de forma síncrona. En el ejemplo anterior se muestra la cantidad de código mínimo necesaria para realizar una transferencia. El objeto `Transfer` puede realizar las transferencias de forma asíncrona y dispone de varias opciones de configuración que puede utilizar para personalizar las transferencias.

Puede cargar los archivos locales a una "subcarpeta" de un bucket de Amazon S3 añadiendo un prefijo de clave al URI `s3://`. En el siguiente ejemplo se cargan los archivos locales en el disco en el bucket `bucket` y se almacenan los archivos con el prefijo de clave `foo`.

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Descarga de un bucket de Amazon S3

Puede descargar de forma recursiva un bucket de Amazon S3 en un directorio local en el disco especificando el argumento `$source` como URI de Amazon S3 (por ejemplo, `s3://bucket`) y el argumento `$dest` como la ruta a un directorio local.

```
// 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

El SDK creará automáticamente los directorios necesarios al descargar los objetos en el bucket.

Puede incluir un prefijo de clave en el URI de Amazon S3 después del bucket para descargar solo los objetos almacenados en una "pseudocarpeta". En el siguiente ejemplo, solo se descargan los archivos almacenados con el prefijo de clave `/foo` del bucket determinado.

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Configuration

El constructor del objeto `Transfer` acepta los siguientes argumentos.

\$client

Es el objeto `Aws\ClientInterface` que hay que utilizar para ejecutar las transferencias.

\$source (string|`Iterator`)

Son los datos de origen que se están transfiriendo. Puede apuntar hacia una ruta local en el disco (por ejemplo, `/path/to/files`) o hacia un bucket de Amazon S3 (por ejemplo, `s3://bucket`). El URI `s3://` también pueden contener un prefijo de clave que se puede utilizar para transferir únicamente los objetos con un prefijo común.

Si el argumento `$source` es un URI de Amazon S3, el argumento `$dest` debe ser un directorio local (y viceversa).

Además de proporcionar un valor de cadena, también puede proporcionar un objeto `\Iterator` que da como resultado nombres de archivo absolutos. Si proporciona un iterador, deberá proporcionar una opción `base_dir` en la matriz asociativa `$options`.

\$dest

Es el destino al que se transfieren los archivos. Si el argumento `$source` es una ruta local en el disco, `$dest` debe ser un URI de bucket de Amazon S3 (por ejemplo, `s3://bucket`). Si el argumento `$source` es un URI de bucket de Amazon S3, el argumento `$dest` debe ser una ruta local en el disco.

\$options

Es una matriz asociativa de [opciones de transferencia](#) (p. 115).

Opciones de transferencia

base_dir (cadena)

Es el directorio base del origen, si `$source` es un iterador. Si la opción `$source` no es una matriz, se omitirá.

before (invocable)

Es una devolución de llamada que se debe invocar antes de cada transferencia. La devolución de llamada debe tener una firma de función como `function (Aws\Command $command) {...}`. El comando proporcionado será un comando `GetObject`, `PutObject`, `CreateMultipartUpload`, `UploadPart` o `CompleteMultipartUpload`.

mup_threshold (int)

El tamaño en bytes por el que debe utilizarse una carga multiparte en lugar de `PutObject`. El valor predeterminado es 16777216 (16 MB).

concurrency (int, valor predeterminado=5)

Es el número de archivos que se debe cargar de forma simultánea. El valor de concurrencia ideal variará en función de la cantidad de archivos que se esté cargando y del tamaño medio de cada archivo. Por lo general, los archivos más pequeños se benefician de una mayor concurrencia, mientras que los archivos más grandes no.

debug (bool)

Establezca este parámetro en `true` para imprimir información de depuración para las transferencias. Establezca un recurso `fopen()` para escribir en un flujo específico en lugar de escribir en `STDOUT`.

Transferencias asíncronas

El objeto `Transfer` es una instancia de `GuzzleHttp\Promise\PromisorInterface`. Esto significa que la transferencia puede producirse de forma asíncrona y que se inicia llamando al método `promise` del objeto.

```
$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!';
});
```

La promesa se rechazará si alguno de los archivos no se puede transferir. Puede administrar la transferencia fallida de forma asíncrona utilizando el método `otherwise` de la promesa. La función `otherwise` acepta una devolución de llamada cuando se produce un error. La devolución de llamada acepta el `$reason` para el rechazo, que por lo general será una instancia de `Aws\Exception` `\AwsException` (aunque se puede enviar cualquier tipo de valor a la devolución de llamada).

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

Dado que el objeto `Transfer` devuelve una promesa, estas transferencias pueden producirse de forma simultánea con otras promesas asíncronas.

Personalización de los comandos del administrador de transferencias

Puede establecer opciones personalizadas en las operaciones ejecutadas por el administrador de transferencias mediante una devolución de llamada transferida a su constructor.

```
$uploader = new Transfer($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';
        }
    },
]);
```

Cifrado del lado del cliente de Amazon S3 con el AWS SDK for PHP versión 3

Con el cifrado en el lado del cliente, los datos se cifran y se descifran directamente en su entorno. Esto significa que estos datos se cifran antes de que se transfieran a Amazon S3 y, por lo tanto, no tendrá que confiar en un servicio externo que gestione el cifrado. Para nuevas implementaciones, sugerimos el uso de `S3EncryptionClientV2` y `S3EncryptionMultipartUploaderV2` sobre las `S3EncryptionClient`

y `S3EncryptionMultipartUploader` obsoletas. Recomendamos que las implementaciones más antiguas sigan utilizando las versiones obsoletas y que intenten migrar. `S3EncryptionClientV2` es compatible con el descifrado de datos que se cifraron con el `S3EncryptionClient` heredado.

El AWS SDK for PHP implementa el [cifrado de sobres](#) y utiliza [OpenSSL](#) para cifrar y descifrar. La implementación es interoperable con [otros SDKs que coinciden con su compatibilidad de características](#). También es compatible con el [flujo de trabajo asíncrono basado en promesas del SDK](#) (p. 66).

Migration Guide

Para aquellos que intentan migrar a desde los clientes obsoletos a los nuevos clientes, hay una guía de migración que se puede encontrar [aquí](#).

Setup

Para comenzar a utilizar el cifrado del lado del cliente, necesita lo siguiente:

- Una [clave de cifrado de AWS KMS](#)
- Un [bucket de S3](#)

Antes de ejecutar cualquier código de ejemplo, configure las credenciales de AWS. Consulte [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44).

Encryption

La carga de un objeto cifrado en `S3EncryptionClientV2` requiere tres parámetros adicionales además de los parámetros `PutObject` estándar:

- `'@KmsEncryptionContext'` es un par clave-valor que se puede utilizar para añadir una capa de seguridad adicional al objeto cifrado. El cliente de cifrado debe pasar la misma clave, lo que hará automáticamente en una llamada para obtener. Si no desea contexto adicional, pase una matriz vacía.
- Los `@CipherOptions` de son configuraciones adicionales para el cifrado, incluido qué cifrado utilizar y tamaño de clave.
- `@MaterialsProvider` es un proveedor que gestiona la generación de una clave de cifrado y un vector de inicialización, así como el cifrado de la clave de cifrado.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV2;

// Let's construct our S3EncryptionClient using an S3Client
$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
```

```

    ]),
    $kmsKeyId
);

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

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

```

Note

Además de los errores de servicio basados en Amazon S3 y AWS KMS, es posible que reciba objetos `InvalidArgumentException` si sus '@CipherOptions' no se configuran correctamente.

Decryption

La descarga y descifrado de un objeto tiene cuatro parámetros adicionales, dos de los cuales son necesarios, además de los parámetros `GetObject` estándar. El cliente detectará las opciones de cifrado básicas por usted.

- **'@SecurityProfile'**: Si se establece en 'V2', solo los objetos cifrados en V2 son compatibles
El formato se puede descifrar. El establecimiento de este parámetro en "V2_AND_LEGACY" también permite descifrar los objetos cifrados en formato compatible con V1. Para admitir la migración, establezca @SecurityProfile en "V2_AND_LEGACY". Utilice "V2" únicamente para el desarrollo de nuevas aplicaciones.
- **'@MaterialsProvider'** es un proveedor que gestiona la generación de una clave de cifrado y un vector de inicialización, como
y cifrar la clave de cifrado.
- **'@KmsAllowDecryptWithAnyCmk'**: (opcional) la configuración de este parámetro en true permite el descifrado
sin proporcionar un ID de clave de KMS al constructor del MaterialsProvider. El valor predeterminado es false.
- **'@CipherOptions'** (opcional) son configuraciones adicionales para el cifrado, incluido el que
Cifrado que se va a utilizar y tamaño de clave.

```

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);

```

```
]);
```

Note

Además de los errores de servicio basados en Amazon S3 y AWS KMS, es posible que reciba objetos `InvalidArgumentException` si sus '@CipherOptions' no se configuran correctamente.

Configuración de Cipher

'Cipher' (cadena)

Es el método Cipher que utiliza el cliente para cifrar. De momento, solo se admite "gcm".

Important

PHP se [actualiza en la versión 7.1](#) para incluir los parámetros adicionales necesarios para [cifrar](#) y [descifrar](#) mediante OpenSSL para el cifrado GCM. Para las versiones 7.0 y anteriores de PHP, se proporciona un relleno para la compatibilidad con GCM que utilizan los clientes de cifrado de `S3EncryptionClientV2` y `S3EncryptionMultipartUploaderV2`. Sin embargo, el rendimiento de las entradas de gran tamaño será mucho más lento utilizando el polígono que utilizando la implementación nativa para PHP 7.1+, por lo que es posible que sea necesario actualizar los entornos de versiones de PHP más antiguas para utilizarlas de forma eficaz.

'KeySize' (int)

Es la longitud de la clave de cifrado del contenido que se genera para el cifrado. La opción por defecto son 256 bits. Las opciones de configuración válidas son 256 y 128 bits.

'Aad' (cadena)

Son datos de autenticación adicionales que se incluyen en la carga cifrada. Esta información se valida al descifrar. Aad solo está disponible cuando se utiliza el cifrado "gcm".

Important

Todos los datos de autenticación adicionales no son compatibles con todos los SDKs de AWS y, por lo tanto, es posible que otros SDKs no puedan descifrar archivos cifrados con este parámetro.

Estrategias de metadatos

También tiene la opción de proporcionar una instancia de una clase que implementa `Aws\Crypto\MetadataStrategyInterface`. Esta interfaz sencilla controla el guardado y la carga del `Aws\Crypto\MetadataEnvelope` que contiene los materiales de cifrado de sobres. El SDK proporciona dos clases que implementan lo siguiente: `Aws\S3\Crypto\HeadersMetadataStrategy` y `Aws\S3\Crypto\InstructionFileMetadataStrategy`. `HeadersMetadataStrategy` se utiliza de forma predeterminada.

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@MetadataStrategy' => $strategy,
    '@KmsEncryptionContext' => [],
    '@CipherOptions' => $cipherOptions,
```

```
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V2',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Las constantes del nombre de la clase de `HeadersMetadataStrategy` y `InstructionFileMetadataStrategy` también se pueden suministrar invocando `::class`.

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

Note

Si se produce un error después cargar un archivo de instrucciones, no se eliminará automáticamente.

Cargas multiparte

También se puede realizar una carga multiparte con el cifrado del lado del cliente. El `Aws\S3\Crypto\S3EncryptionMultipartUploaderV2` prepara la secuencia de origen para el cifrado antes de cargarla. La creación de una incluye una experiencia similar al uso de la `Aws\S3\MultipartUploader` y la `Aws\S3\Crypto\S3EncryptionClientV2`. El `S3EncryptionMultipartUploaderV2` puede gestionar la misma opción `'@MetadataStrategy'` que el `S3EncryptionClientV2`, así como todas las configuraciones de `'@CipherOptions'` disponibles.

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyId
);

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

$multipartUploader = new S3EncryptionMultipartUploaderV2(
    new S3Client([
```

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

Note

Además de los errores de servicio basados en Amazon S3 y AWS KMS, es posible que reciba objetos `InvalidArgumentException` si sus '@CipherOptions' no se configuran correctamente.

Ejemplos de código del AWS SDK for PHP versión 3

El AWS SDK for PHP versión 3 incluye ejemplos de código que muestran escenarios comunes de Amazon Web Services que utilizan el SDK.

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Temas

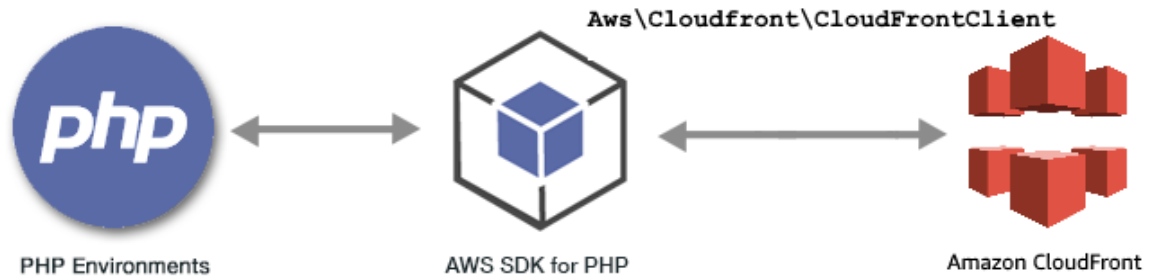
- [Ejemplos de Amazon CloudFront que utilizan el AWS SDK for PHP versión 3 \(p. 122\)](#)
- [Firma de solicitudes personalizadas del dominio de Amazon CloudSearch con el AWS SDK for PHP versión 3 \(p. 142\)](#)
- [Ejemplos de Amazon CloudWatch que utilizan el AWS SDK for PHP versión 3 \(p. 144\)](#)
- [Ejemplos de Amazon EC2 que utilizan el AWS SDK for PHP versión 3 \(p. 161\)](#)
- [Firma de una petición de búsqueda de Amazon Elasticsearch Service con el AWS SDK for PHP versión 3 \(p. 171\)](#)
- [Ejemplos de AWS IAM que utilizan el AWS SDK for PHP versión 3 \(p. 172\)](#)
- [Ejemplos de AWS Key Management Service que utilizan el AWS SDK for PHP versión 3 \(p. 190\)](#)
- [Ejemplos de Amazon Kinesis que utilizan el AWS SDK for PHP versión 3 \(p. 207\)](#)
- [Ejemplos de AWS Elemental MediaConvert que utilizan el AWS SDK for PHP versión 3 \(p. 219\)](#)
- [Ejemplos de Amazon S3 que utilizan el AWS SDK for PHP versión 3 \(p. 227\)](#)
- [Administración de secretos mediante la API de Secrets Manager y el AWS SDK for PHP versión 3 \(p. 248\)](#)
- [Ejemplos de Amazon SES que utilizan el AWS SDK for PHP versión 3 \(p. 254\)](#)
- [Ejemplos de Amazon SNS que utilizan el AWS SDK for PHP versión 3 \(p. 278\)](#)
- [Ejemplos de Amazon SQS que utilizan el AWS SDK for PHP versión 3 \(p. 292\)](#)

Ejemplos de Amazon CloudFront que utilizan el AWS SDK for PHP versión 3

Amazon CloudFront es un servicio web de AWS que agiliza la distribución de contenido web estático y dinámico desde su propio servidor web o desde un servidor de AWS, como por ejemplo, Amazon S3. CloudFront ofrece contenido a través de una red mundial de centros de datos denominados ubicaciones de

borde. Cuando un usuario solicita contenido que usted distribuye con CloudFront, se le dirige a la ubicación de borde que ofrece la latencia más baja. Si el contenido todavía no está en la memoria caché de dicha ubicación, CloudFront recupera una copia desde el servidor de origen, se la proporciona al usuario y, a continuación, la almacena en caché para futuras solicitudes.

Para obtener más información acerca de CloudFront, consulte la [Guía para desarrolladores de Amazon CloudFront](#).



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- [Administración de distribuciones de Amazon CloudFront mediante la API de CloudFront y AWS SDK for PHP versión 3](#) (p. 123)
- [Administración de invalidaciones de Amazon CloudFront mediante la API de CloudFront y el AWS SDK for PHP versión 3](#) (p. 133)
- [Firma de Amazon CloudFront URLs con el AWS SDK for PHP versión 3](#) (p. 137)

Administración de distribuciones de Amazon CloudFront mediante la API de CloudFront y AWS SDK for PHP versión 3

Amazon CloudFront almacena en caché el contenido en ubicaciones de borde en todo el mundo para acelerar la distribución de los archivos estáticos y dinámicos que usted almacena en su propio servidor o en un servicio de Amazon como Amazon S3 o Amazon EC2. Cuando los usuarios solicitan contenido de su sitio web, CloudFront lo envía desde la ubicación de borde más cercana, si el archivo está almacenado en caché en ella. De lo contrario CloudFront recupera una copia del archivo, lo envía y, a continuación, lo almacena en caché para la siguiente solicitud. El almacenamiento en caché de contenido en una ubicación de borde reduce la latencia de las solicitudes similares de los usuarios de dicha área.

Para cada distribución de CloudFront que se crea, es preciso especificar dónde se guarda el contenido y cómo distribuirlo cuando los usuarios lo solicitan. Este tema se centra en las distribuciones de archivos estáticos y dinámicos como HTML, CSS, JSON y archivos de imágenes. Para obtener información sobre el uso de CloudFront con vídeo bajo demanda, consulte [Vídeo bajo demanda y en streaming con CloudFront](#).

Los siguientes ejemplos muestran cómo:

- Describe cómo crear una distribución con [CreateDistribution](#).
- Obtenga una distribución con [GetDistribution](#).
- Obtener una lista de distribuciones con [ListDistributions](#).
- Actualice distribuciones con [UpdateDistributions](#).
- Deshabilite distribuciones con [DisableDistribution](#).

- Elimine distribuciones con [DeleteDistributions](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información sobre el uso de Amazon CloudFront, consulte la [Guía para desarrolladores de Amazon CloudFront](#).

Creación de una distribución de CloudFront

Puede crear una distribución de un bucket de Amazon S3. En el siguiente ejemplo, los parámetros opcionales están comentados, pero se muestran los valores predeterminados. Para añadir personalizaciones a la distribución, anule el comentario del valor y del parámetro dentro de `$distribution`.

Para crear una distribución de CloudFront, utilice la operación [CreateDistribution](#).

Importaciones

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

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

Código de muestra

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);

        $message = '';

        if (isset($result['Distribution']['Id']))
        {
            $message = 'Distribution created with the ID of ' .
                $result['Distribution']['Id'];
        }

        $message .= ' and an effective URI of ' .
            $result['@metadata']['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'my-bucket-name.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
```

```

$defaultCacheBehavior = [
    'AllowedMethods' => [
        'CachedMethods' => [
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Items' => ['HEAD', 'GET'],
        'Quantity' => 2
    ],
    'Compress' => false,
    'DefaultTTL' => 0,
    'FieldLevelEncryptionId' => '',
    'ForwardedValues' => [
        'Cookies' => [
            'Forward' => 'none'
        ],
        'Headers' => [
            'Quantity' => 0
        ],
        'QueryString' => false,
        'QueryStringCacheKeys' => [
            'Quantity' => 0
        ]
    ],
    'LambdaFunctionAssociations' => ['Quantity' => 0],
    'MaxTTL' => 0,
    'MinTTL' => 0,
    'SmoothStreaming' => false,
    'TargetOriginId' => $originName,
    'TrustedSigners' => [
        'Enabled' => false,
        'Quantity' => 0
    ],
    'ViewerProtocolPolicy' => 'allow-all'
];
$enabled = false;
$origin = [
    'Items' => [
        [
            'DomainName' => $s3BucketURL,
            'Id' => $originName,
            'OriginPath' => '',
            'CustomHeaders' => ['Quantity' => 0],
            'S3OriginConfig' => ['OriginAccessIdentity' => '']
        ]
    ],
    'Quantity' => 1
];
$distribution = [
    'CallerReference' => $callerReference,
    'Comment' => $comment,
    'DefaultCacheBehavior' => $defaultCacheBehavior,
    'Enabled' => $enabled,
    'Origins' => $origin
];

$cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-1'
]);

echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.

```

```
// createsTheS3Distribution();
```

Recuperación de una distribución de CloudFront

Para recuperar el estado y los detalles de una distribución de CloudFront especificada, utilice la operación [GetDistribution](#).

Importaciones

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

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

Código de muestra

```
function getDistribution($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId
        ]);

        $message = '';

        if (isset($result['Distribution']['Status']))
        {
            $message = 'The status of the distribution with the ID of ' .
                $result['Distribution']['Id'] . ' is currently ' .
                $result['Distribution']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri']))
        {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get the specified distribution. ' .
                'The distribution\'s status is not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getsADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getDistribution($cloudFrontClient, $distributionId);
}

// Uncomment the following line to run this code in an AWS account.
```

```
// getsADistribution();
```

Listado de distribuciones de CloudFront

Obtenga una lista de las distribuciones de CloudFront existentes en la región de AWS especificada desde su cuenta actual con la operación [ListDistributions](#).

Importaciones

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

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

Código de muestra

```
function listDistributions($cloudFrontClient)
{
    try {
        $result = $cloudFrontClient->listDistributions([]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

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

    $distributions = listDistributions($cloudFrontClient);

    if (count($distributions) == 0)
    {
        echo 'Could not find any distributions.';
    } else {
        foreach ($distributions['DistributionList']['Items'] as $distribution)
        {
            echo 'The distribution with the ID of ' . $distribution['Id'] .
                ' has the status of ' . $distribution['Status'] . ' . ' . "\n";
        }
    }
}

// Uncomment the following line to run this code in an AWS account.
// listTheDistributions();
```

Actualización de una distribución de CloudFront

La actualización de una distribución de CloudFront es similar a la creación de una distribución. Sin embargo, cuando se actualiza una distribución, se necesitan más campos y deben incluirse todos los valores. Para realizar cambios en una distribución existente, le recomendamos que primero recupere la distribución y que actualice los valores que desee cambiar en la matriz `$distribution`.

Para actualizar una distribución de CloudFront determinada, utilice la operación [UpdateDistribution](#).

Importaciones

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

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

Código de muestra

```
function updateDistribution($cloudFrontClient, $distributionId,
    $distributionConfig, $eTag)
{
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);

        return 'The distribution with the following effective URI has ' .
            'been updated: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig']))
        {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag']))
        {
            return [
                'ETag' => $result['ETag'],
            ];
        }
    }
}
```

```
        'effectiveUri' => $result['@metadata']['effectiveUri']
    ];
} else {
    return [
        'Error' => 'Error: Cannot find distribution ETag header value.',
        'effectiveUri' => $result['@metadata']['effectiveUri']
    ];
}

} catch (AwsException $e) {
    return [
        'Error' => 'Error: ' . $e->getAwsErrorMessage()
    ];
}
}

function updateADistribution()
{
    // $distributionId = 'E1BTGP2EXAMPLE';
    $distributionId = 'E1X3BKQ569KEMH';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To change a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To change a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration.
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    // To change a distribution's configuration, you can set the
    // distribution's related configuration value as part of a change request,
    // for example:
    // 'Enabled' => true
    // Some configuration values are required to be specified as part of a change
    // request, even if you don't plan to change their values. For ones you
    // don't want to change but are required to be specified, you can just reuse
    // their current values, as follows.
    $distributionConfig = [
        'CallerReference' => $currentConfig['DistributionConfig']['CallerReference'],
        'Comment' => $currentConfig['DistributionConfig']['Comment'],
        'DefaultCacheBehavior' => $currentConfig['DistributionConfig']
["DefaultCacheBehavior"],
        'DefaultRootObject' => $currentConfig['DistributionConfig']["DefaultRootObject"],
        'Enabled' => $currentConfig['DistributionConfig']["Enabled"],
        'Origins' => $currentConfig['DistributionConfig']["Origins"],
        'Aliases' => $currentConfig['DistributionConfig']["Aliases"],
        'CustomErrorResponses' => $currentConfig['DistributionConfig']
["CustomErrorResponses"],
        'HttpVersion' => $currentConfig['DistributionConfig']["HttpVersion"],
        'CacheBehaviors' => $currentConfig['DistributionConfig']["CacheBehaviors"],
        'Logging' => $currentConfig['DistributionConfig']["Logging"],
```



```
'PriceClass' => $currentConfig['DistributionConfig']['PriceClass'],
'Restrictions' => $currentConfig['DistributionConfig']['Restrictions'],
'ViewerCertificate' => $currentConfig['DistributionConfig']['ViewerCertificate'],
'WebACLId' => $currentConfig['DistributionConfig']['WebACLId']
];

echo updateDistribution($cloudFrontClient, $distributionId,
    $distributionConfig, $eTag['ETag']);
}

// Uncomment the following line to run this code in an AWS account.
// updateADistribution();
```

Deshabilitación de una distribución de CloudFront

Para desactivar o eliminar una distribución, cambie su estado de implementada a deshabilitada.

Para deshabilitar la distribución de CloudFront especificada, utilice la operación [DisableDistribution](#).

Importaciones

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

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

Código de muestra

```
function disableDistribution($cloudFrontClient, $distributionId,
    $distributionConfig, $eTag)
{
    try {
        $result = $cloudFrontClient->updateDistribution([
            'DistributionConfig' => $distributionConfig,
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution with the following effective URI has ' .
            'been disabled: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionConfig($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['Distribution']['DistributionConfig']))
        {
            return [
                'DistributionConfig' => $result['Distribution']['DistributionConfig'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution configuration details.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    }
}
```

```
    }

    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag']))
        {
            return [
                'ETag' => $result['ETag'],
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        } else {
            return [
                'Error' => 'Error: Cannot find distribution ETag header value.',
                'effectiveUri' => $result['@metadata']['effectiveUri']
            ];
        }
    } catch (AwsException $e) {
        return [
            'Error' => 'Error: ' . $e->getAwsErrorMessage()
        ];
    }
}

function disableADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To disable a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    }

    // To delete a distribution, you must also first get information about
    // the distribution's current configuration. Then you must use that
    // information to build a new configuration, including setting the new
    // configuration to disabled.
    $currentConfig = getDistributionConfig($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $currentConfig)) {
        exit($currentConfig['Error']);
    }

    $distributionConfig = [
```

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

echo disableDistribution($cloudFrontClient, $distributionId,
    $distributionConfig, $eTag['ETag']);
}

// Uncomment the following line to run this code in an AWS account.
// disableADistribution();
```

Eliminación de una distribución de CloudFront

Después de deshabilitar una distribución, puede eliminarla.

Para eliminar una distribución de CloudFront especificada, utilice la operación [DeleteDistribution](#).

Importaciones

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

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

Código de muestra

```
function deleteDistribution($cloudFrontClient, $distributionId, $eTag)
{
    try {
        $result = $cloudFrontClient->deleteDistribution([
            'Id' => $distributionId,
            'IfMatch' => $eTag
        ]);
        return 'The distribution at the following effective URI has ' .
            'been deleted: ' . $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getDistributionETag($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId,
        ]);

        if (isset($result['ETag']))
```

```
{
    return [
        'ETag' => $result['ETag'],
        'effectiveUri' => $result['@metadata']['effectiveUri']
    ];
} else {
    return [
        'Error' => 'Error: Cannot find distribution ETag header value.',
        'effectiveUri' => $result['@metadata']['effectiveUri']
    ];
}

} catch (AwsException $e) {
    return [
        'Error' => 'Error: ' . $e->getAwsErrorMessage()
    ];
}
}

function deleteADistribution()
{
    $distributionId = 'E17G7YNEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    // To delete a distribution, you must first get the distribution's
    // ETag header value.
    $eTag = getDistributionETag($cloudFrontClient, $distributionId);

    if (array_key_exists('Error', $eTag)) {
        exit($eTag['Error']);
    } else {
        echo deleteDistribution($cloudFrontClient, $distributionId,
            $eTag['ETag']);
    }
}

// Uncomment the following line to run this code in an AWS account.
// deleteADistribution();
```

Administración de invalidaciones de Amazon CloudFront mediante la API de CloudFront y el AWS SDK for PHP versión 3

Amazon CloudFront almacena en caché copias de archivos estáticos y dinámicos en ubicaciones de borde en todo el mundo. Para eliminar o actualizar un archivo en todas las ubicaciones de borde, cree una invalidación para cada archivo o para un grupo de archivos.

Cada mes, sus primeras 1000 invalidaciones son gratuitas. Para obtener más información acerca de cómo eliminar contenido de una ubicación de borde de CloudFront, consulte [Invalidar archivos](#).

Los siguientes ejemplos muestran cómo:

- Describe cómo crear una invalidación de una distribución con [CreateInvalidation](#).
- Obtenga una invalidación de una distribución con [GetInvalidation](#).
- Obtener una lista de distribuciones con [ListInvalidations](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información sobre el uso de Amazon CloudFront, consulte la [Guía para desarrolladores de Amazon CloudFront](#).

Creación de una invalidación de una distribución

Para crear una invalidación de una distribución de CloudFront, especifique la ubicación de la ruta de los archivos que necesita eliminar. Este ejemplo invalida todos los archivos de la distribución, pero puede identificar archivos específicos en `Items`.

Para crear una invalidación de una distribución de CloudFront, utilice la operación [CreateInvalidation](#).

Importaciones

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

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

Código de muestra

```
function createInvalidation($cloudFrontClient, $distributionId,
    $callerReference, $paths, $quantity)
{
    try {
        $result = $cloudFrontClient->createInvalidation([
            'DistributionId' => $distributionId,
            'InvalidationBatch' => [
                'CallerReference' => $callerReference,
                'Paths' => [
                    'Items' => $paths,
                    'Quantity' => $quantity,
                ],
            ],
        ]);

        $message = '';

        if (isset($result['Location']))
        {
            $message = 'The invalidation location is: ' .
                $result['Location'];
        }

        $message .= ' and the effective URI is ' .
            $result['@metadata']['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheInvalidation()
{
}
```

```
$distributionId = 'E1WICG14DUW2AP'; // 'E17G7YNEXAMPLE';
$callerReference = 'my-unique-value';
$paths = ['/*'];
$quantity = 1;

$cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
    'profile' => 'default',
    'version' => '2018-06-18',
    'region' => 'us-east-1'
]);

echo createInvalidation($cloudFrontClient, $distributionId,
    $callerReference, $paths, $quantity);
}

// Uncomment the following line to run this code in an AWS account.
// createTheInvalidation();
```

Obtención de una invalidación de una distribución

Para recuperar el estado y los detalles acerca de una invalidación de una distribución de CloudFront, utilice la operación [GetInvalidation](#).

Importaciones

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

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

Código de muestra

```
function getInvalidation($cloudFrontClient, $distributionId, $invalidationId)
{
    try {
        $result = $cloudFrontClient->getInvalidation([
            'DistributionId' => $distributionId,
            'Id' => $invalidationId,
        ]);

        $message = '';

        if (isset($result['Invalidation']['Status']))
        {
            $message = 'The status for the invalidation with the ID of ' .
                $result['Invalidation']['Id'] . ' is ' .
                $result['Invalidation']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri']))
        {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get information about ' .
                'the invalidation. The invalidation\'s status ' .
                'was not available.';
        }

        return $message;
    } catch (AwsException $e) {
```

```
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getsAnInvalidation()
{
    $distributionId = 'E1BTGP2EXAMPLE';
    $invalidationId = 'I1CDEZZEXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getInvalidation($cloudFrontClient, $distributionId, $invalidationId);
}

// Uncomment the following line to run this code in an AWS account.
// getsAnInvalidation();
```

Listado de las invalidaciones de una distribución

Para enumerar todas las invalidaciones de distribución de CloudFront actuales, utilice la operación [ListInvalidations](#).

Importaciones

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

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

Código de muestra

```
function listInvalidations($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->listInvalidations([
            'DistributionId' => $distributionId
        ]);
        return $result;
    } catch (AwsException $e) {
        exit('Error: ' . $e->getAwsErrorMessage());
    }
}

function listTheInvalidations()
{
    $distributionId = 'E1WICG1EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $invalidations = listInvalidations($cloudFrontClient,
        $distributionId);

    if (isset($invalidations['InvalidationList']))
    {
```

```
if ($invalidations['InvalidationList']['Quantity'] > 0)
{
    foreach ($invalidations['InvalidationList']['Items'] as $invalidation)
    {
        echo 'The invalidation with the ID of ' . $invalidation['Id'] .
            ' has the status of ' . $invalidation['Status'] . ' . ' . "\n";
    }
} else {
    echo 'Could not find any invalidations for the specified distribution.';
}
} else {
    echo 'Error: Could not get invalidation information. Could not get ' .
        'information about the specified distribution.';
}
}

// Uncomment the following line to run this code in an AWS account.
// listTheInvalidations();
```

Firma de Amazon CloudFront URLs con el AWS SDK for PHP versión 3

Los URLs firmados le permiten proporcionar a los usuarios acceso a su contenido privado. Una URL firmada incluye información adicional (por ejemplo, una fecha y hora de vencimiento), que le proporciona un mayor control sobre el acceso a su contenido. Esta información adicional aparece en una instrucción de política basada en una política predefinida o personalizada. Para obtener información acerca de cómo configurar distribuciones privadas y por qué es necesario firmar URLs, consulte [Distribución de contenido privado a través de Amazon CloudFront en la](#) .Amazon CloudFront Developer Guide

- Cree una URL de Amazon CloudFront firmada mediante [getSignedURL](#).
- Describe cómo crear una cookie de Amazon CloudFront firmada mediante [getSignedCookie](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información sobre el uso de Amazon CloudFront, consulte la [Guía para desarrolladores de Amazon CloudFront](#).

Firma de CloudFront URLs para distribuciones privadas

Para firmar una URL, puede usar el cliente de CloudFront del SDK. En primer lugar, debe crear un objeto `CloudFrontClient`. Para firmar una URL de CloudFront para un recurso de vídeo, puede usar una política predefinida o una personalizada.

Importaciones

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

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

Código de muestra


```
function signPrivateDistribution($cloudFrontClient, $resourceKey, $expires,
    $privateKey, $keyPairId)
{
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;

    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistribution()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2014-11-06',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistribution($cloudFrontClient, $resourceKey, $expires,
        $privateKey, $keyPairId);
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistribution();
```

Usar una política personalizada al crear CloudFront URLs

Para utilizar una política personalizada, indique la clave `policy` en lugar de `expires`.

Importaciones

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

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

Código de muestra

```
function signPrivateDistributionPolicy($cloudFrontClient, $resourceKey,
    $customPolicy, $privateKey, $keyPairId)
{
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

    };
```

```
        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistributionPolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2014-11-06',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistributionPolicy($cloudFrontClient, $resourceKey,
        $customPolicy, $privateKey, $keyPairId);
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistributionPolicy();
```

Uso de una URL firmada de CloudFront

El formato de la URL firmada depende de si la URL que se está firmando utiliza el esquema "HTTP" o "RTMP". En el caso de "HTTP", se devuelve la URL absoluta completa. Para "RTMP" solo se devuelve la URL relativa para mayor comodidad. Esto se debe a que algunos reproductores requieren que el host y la ruta se indiquen como parámetros independientes.

El siguiente ejemplo muestra cómo puede utilizar la URL firmada para construir una página web que muestra un vídeo utilizando [JWPlayer](#). Se aplicaría el mismo tipo de técnica a otros reproductores como [FlowPlayer](#), pero se requeriría un código diferente en el cliente.

```
<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>
```

Firmar cookies de CloudFront para distribuciones privadas

Como alternativa a las URLs firmadas, también puede conceder a los clientes acceso a una distribución privada a través de cookies firmadas. Las cookies firmadas permiten ofrecer acceso a múltiples archivos restringidos, por ejemplo a todos los archivos de vídeo en formato HLS, o a todos los archivos del área de suscriptores de un sitio web. Para obtener más información sobre por qué es posible que desee utilizar cookies firmadas en lugar de URLs firmadas (o viceversa), consulte [Elegir entre cookies firmadas URLs y firmadas](#) en la Amazon CloudFront Developer Guide.

Crear una cookie firmada es similar a crear una URL firmada. La única diferencia es el método al que se llama (getSignedCookie en lugar de getSignedUrl).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\CloudFront\CloudFrontClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
function signCookie($cloudFrontClient, $resourceKey, $expires,  
    $privateKey, $keyPairId)  
{  
    try {  
        $result = $cloudFrontClient->getSignedCookie([  
            'url' => $resourceKey,  
            'expires' => $expires,  
            'private_key' => $privateKey,  
            'key_pair_id' => $keyPairId  
        ]);  
  
        return $result;  
  
    } catch (AwsException $e) {  
        return [ 'Error' => $e->getAwsErrorMessage() ];  
    }  
}  
  
function signACookie()  
{  
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';  
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.  
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';  
    $keyPairId = 'AAPKAJIKZATYYEXAMPLE';  
  
    $cloudFrontClient = new CloudFrontClient([  
        'profile' => 'default',  
        'version' => '2014-11-06',  
        'region' => 'us-east-1'  
    ]);  
  
    $result = signCookie($cloudFrontClient, $resourceKey, $expires,  
        $privateKey, $keyPairId);  
  
    /* If successful, returns something like:
```

```
CloudFront-Expires = 1589926678
CloudFront-Signature = Lv1DyC2q...2HPXaQ__
CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
*/
foreach($result as $key => $value)
{
    echo $key . ' = ' . $value . "\n";
}

// Uncomment the following line to run this code in an AWS account.
// signACookie();
```

Uso de una política personalizada al crear cookies de CloudFront

Al igual que con `getSignedUrl`, puede especificar un parámetro `'policy'` en lugar de un parámetro `expires` y un parámetro `url` para firmar una cookie con una política personalizada. La política personalizada puede contener comodines en la clave de recurso. Esto le permite crear una sola cookie firmada para varios archivos.

`getSignedCookie` devuelve una matriz de pares clave-valor, que deben configurarse como cookies en su totalidad para obtener acceso a una distribución privada.

Importaciones

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

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

Código de muestra

```
function signCookiePolicy($cloudFrontClient, $customPolicy,
    $privateKey, $keyPairId)
{
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;

    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}

function signACookiePolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
```

```
    }  
  ]  
}  
POLICY;  
$privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';  
$keyPairId = 'AAPKAJIKZATYYYEXAMPLE';  
  
$cloudFrontClient = new CloudFrontClient([  
    'profile' => 'default',  
    'version' => '2014-11-06',  
    'region' => 'us-east-1'  
]);  
  
$result = signCookiePolicy($cloudFrontClient, $customPolicy,  
    $privateKey, $keyPairId);  
  
/* If successful, returns something like:  
CloudFront-Policy = eyJTdGF0...fXl9XX0_  
CloudFront-Signature = RowqEQWZ...N8vetw__  
CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE  
*/  
foreach ($result as $key => $value) {  
    echo $key . ' = ' . $value . "\n";  
}  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// signACookiePolicy();
```

Envío de cookies de CloudFront al cliente Guzzle

También puede pasar estas cookies a una `GuzzleHttp\Cookie\CookieJar` para su uso con un cliente Guzzle.

```
use GuzzleHttp\Client;  
use GuzzleHttp\Cookie\CookieJar;  
  
$distribution = "example-distribution.cloudfront.net";  
$client = new \GuzzleHttp\Client([  
    'base_uri' => "https://$distribution",  
    'cookies' => CookieJar::fromArray($signedCookieCustomPolicy, $distribution),  
]);  
  
$client->get('video.mp4');
```

Para obtener más información, consulte [Uso de cookies firmadas](#) en la Amazon CloudFront Developer Guide.

Firma de solicitudes personalizadas del dominio de Amazon CloudSearch con el AWS SDK for PHP versión 3

Las solicitudes del dominio Amazon CloudSearch se pueden personalizar además de la compatibilidad de AWS SDK for PHP. En los casos en que necesite realizar solicitudes personalizadas a dominios protegidos por la autenticación de IAM, puede utilizar los proveedores de credenciales del SDK y sus signatarios para firmar cualquier [solicitud de PSR-7](#).

Por ejemplo, si está siguiendo la [Guía de introducción de Cloud Search](#) y desea utilizar un dominio protegido por IAM en el [Paso 3](#), tendría que firmar y ejecutar su solicitud de la siguiente manera.

Los siguientes ejemplos muestran cómo:

- Firme una solicitud con el protocolo de firma de AWS mediante [SignatureV4](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Firma de una solicitud de dominio deCSlong

Importaciones

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

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

Código de muestra

```
function searchDomain($client, $domainName, $domainId,
    $domainRegion, $searchString)
{
    $domainPrefix = 'search-';
    $cloudSearchDomain = 'cloudsearch.amazonaws.com';
    $cloudSearchVersion = '2013-01-01';
    $searchPrefix = 'search?';

    // Specify the search to send.
    $request = new Request(
        'GET',
        "https://{ $domainPrefix } { $domainName } - { $domainId } . { $domainRegion } . " .
            "{ $cloudSearchDomain } / { $cloudSearchVersion } / " .
            "{ $searchPrefix } { $searchString } "
    );

    // Get default AWS account access credentials.
    $credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

    // Sign the search request with the credentials.
    $signer = new SignatureV4('cloudsearch', $domainRegion);
    $request = $signer->signRequest($request, $credentials);

    // Send the signed search request.
    $response = $client->send($request);

    // Report the search results, if any.
    $results = json_decode($response->getBody());

    $message = '';

    if ($results->hits->found > 0) {
```

```
        $message .= 'Search results:' . "\n";

        foreach($results->hits->hit as $hit)
        {
            $message .= $hit->fields->title . "\n";
        }
    } else {
        $message .= 'No search results.';
    }

    return $message;
}

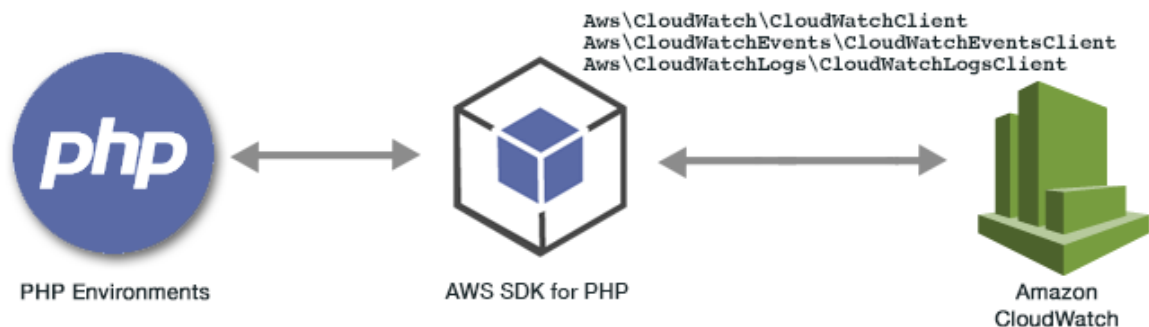
function searchADomain()
{
    $domainName = 'my-search-domain';
    $domainId = '7kbitd6nyiglhtdmssxEXAMPLE';
    $domainRegion = 'us-east-1';
    $searchString = 'q=star+wars&return=title';
    $client = new Client();

    echo searchDomain($client, $domainName, $domainId,
        $domainRegion, $searchString);
}

// Uncomment the following line to run this code in an AWS account.
// searchADomain();
```

Ejemplos de Amazon CloudWatch que utilizan el AWS SDK for PHP versión 3

Amazon CloudWatch (CloudWatch) es un servicio web que monitorea los recursos y las aplicaciones de Amazon Web Services (AWS) que ejecuta en AWS en tiempo real. Puede utilizar CloudWatch para recopilar y hacer un seguimiento de métricas, que son las variables que puede medir en sus recursos y aplicaciones. Las alarmas de CloudWatch envían notificaciones o realizan cambios automáticamente en los recursos que está supervisando en función de las reglas que defina.



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Temas

- [Uso de alarmas de Amazon CloudWatch con el AWS SDK for PHP versión 3 \(p. 145\)](#)
- [Obtención de métricas de Amazon CloudWatch con el AWS SDK for PHP versión 3 \(p. 148\)](#)
- [Publicación de métricas personalizadas en Amazon CloudWatch con el AWS SDK for PHP versión 3 \(p. 153\)](#)
- [Envío de eventos a Amazon CloudWatch Events con el AWS SDK for PHP versión 3 \(p. 156\)](#)
- [Uso de acciones de alarma con las alarmas de Amazon CloudWatch con el AWS SDK for PHP versión 3 \(p. 159\)](#)

Uso de alarmas de Amazon CloudWatch con el AWS SDK for PHP versión 3

Una alarma de Amazon CloudWatch vigila una métrica determinada durante el periodo especificado. Realiza una o varias acciones según el valor de la métrica con respecto a un umbral dado durante varios periodos de tiempo.

Los siguientes ejemplos muestran cómo:

- Describir una alarma utilizando [DescribeAlarms](#).
- Crear una alarma utilizando [PutMetricAlarm](#).
- Eliminar una alarma utilizando [DeleteAlarms](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Describir alarmas

Importaciones

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

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

Código de muestra

```
function describeAlarms($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->describeAlarms();

        $message = '';

        if (isset($result['@metadata']['effectiveUri']))
        {
            $message .= 'Alarms at the effective URI of ' .
                $result['@metadata']['effectiveUri'] . "\n\n";
        }
    }
}
```



```
        if (isset($result['CompositeAlarms']))
        {
            $message .= "Composite alarms:\n";

            foreach ($result['CompositeAlarms'] as $alarm) {
                $message .= $alarm['AlarmName'] . "\n";
            }
        } else {
            $message .= "No composite alarms found.\n";
        }

        if (isset($result['MetricAlarms']))
        {
            $message .= "Metric alarms:\n";

            foreach ($result['MetricAlarms'] as $alarm) {
                $message .= $alarm['AlarmName'] . "\n";
            }
        } else {
            $message .= 'No metric alarms found.';
        }
    } else {
        $message .= 'No alarms found.';
    }

    return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}

}

function describeTheAlarms()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarms($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarms();
```

Crear una alarma

Importaciones

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

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

Código de muestra

```
function putMetricAlarm($cloudWatchClient, $cloudWatchRegion,
    $alarmName, $namespace, $metricName,
    $dimensions, $statistic, $period, $comparison, $threshold,
    $evaluationPeriods)
{
```

```
try {
    $result = $cloudWatchClient->putMetricAlarm([
        'AlarmName' => $alarmName,
        'Namespace' => $namespace,
        'MetricName' => $metricName,
        'Dimensions' => $dimensions,
        'Statistic' => $statistic,
        'Period' => $period,
        'ComparisonOperator' => $comparison,
        'Threshold' => $threshold,
        'EvaluationPeriods' => $evaluationPeriods
    ]);

    if (isset($result['@metadata']['effectiveUri']))
    {
        if ($result['@metadata']['effectiveUri'] ==
            'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com')
        {
            return 'Successfully created or updated specified alarm.';
        } else {
            return 'Could not create or update specified alarm.';
        }
    } else {
        return 'Could not create or update specified alarm.';
    }
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}

}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanOrEqualToThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);
}
```

```
        echo putMetricAlarm($cloudWatchClient, $cloudWatchRegion,
            $alarmName, $namespace, $metricName,
            $dimensions, $statistic, $period, $comparison, $threshold,
            $evaluationPeriods);
    }

    // Uncomment the following line to run this code in an AWS account.
    // putTheMetricAlarm();
```

Eliminar alarmas

Importaciones

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

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

Código de muestra

```
function deleteAlarms($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->deleteAlarms([
            'AlarmNames' => $alarmNames
        ]);

        return 'The specified alarms at the following effective URI have ' .
            'been deleted or do not currently exist: ' .
            $result['@metadata']['effectiveUri'];

    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function deleteTheAlarms()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo deleteAlarms($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// deleteTheAlarms();
```

Obtención de métricas de Amazon CloudWatch con el AWS SDK for PHP versión 3

Las métricas son los datos sobre el desempeño de los sistemas. También puede habilitar el monitoreo detallado de algunos recursos como las instancias Amazon EC2 o de sus propias métricas de aplicación.

Los siguientes ejemplos muestran cómo:

- Obtenga una lista de métricas mediante [ListMetrics](#).
- Recuperar alarmas para una métrica utilizando [DescribeAlarmsForMetric](#).
- Obtenga estadísticas para una métrica especificada utilizando [GetMetricStatistics](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Listar métricas

Importaciones

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

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

Código de muestra

```
function listMetrics($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->listMetrics();

        $message = '';

        if (isset($result['@metadata']['effectiveUri']))
        {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if ((isset($result['Metrics'])) and
                (count($result['Metrics']) > 0))
            {
                $message .= "Metrics found:\n\n";

                foreach($result['Metrics'] as $metric)
                {
                    $message .= 'For metric ' . $metric['MetricName'] .
                        ' in namespace ' . $metric['Namespace'] . ":\n";

                    if ((isset($metric['Dimensions'])) and
                        (count($metric['Dimensions']) > 0))
                    {
                        $message .= "Dimensions:\n";

                        foreach ($metric['Dimensions'] as $dimension)
                        {
                            $message .= 'Name: ' . $dimension['Name'] .
                                ', Value: ' . $dimension['Value'] . "\n";
                        }

                        $message .= "\n";
                    } else {
                        $message .= "No dimensions.\n\n";
                    }
                }
            }
        }
    } catch (AwsException $e) {
        // handle error
    }
}
```

```
        }
    } else {
        $message .= 'No metrics found.';
    }
} else {
    $message .= 'No metrics found.';
}

return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
}

function listTheMetrics()
{
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo listMetrics($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// listTheMetrics();
```

Recuperar alarmas para una métrica

Importaciones

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

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

Código de muestra

```
function describeAlarmsForMetric($cloudWatchClient, $metricName,
    $namespace, $dimensions)
{
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,
            'Namespace' => $namespace,
            'Dimensions' => $dimensions
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri']))
        {
            $message .= 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if ((isset($result['MetricAlarms'])) and
                (count($result['MetricAlarms']) > 0))
            {
                $message .= 'Matching alarms for ' . $metricName . ":\n\n";
            }
        }
    }
}
```

```
        foreach ($result['MetricAlarms'] as $alarm)
        {
            $message .= $alarm['AlarmName'] . "\n";
        }
    } else {
        $message .= 'No matching alarms found for ' . $metricName . '.';
    }
} else {
    $message .= 'No matching alarms found for ' . $metricName . '.';
}

return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value'=> 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'my-bucket'
        ]
    ];

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarmsForMetric($cloudWatchClient, $metricName,
        $namespace, $dimensions);
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

Obtener estadísticas de métricas

Importaciones

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

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

Código de muestra

```
function getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit)
{
    try {
        $result = $cloudWatchClient->getMetricStatistics([
            'Namespace' => $namespace,
```

```
        'MetricName' => $metricName,
        'Dimensions' => $dimensions,
        'StartTime' => $startTime,
        'EndTime' => $endTime,
        'Period' => $period,
        'Statistics' => $statistics,
        'Unit' => $unit
    ]]);

    $message = '';

    if (isset($result['@metadata']['effectiveUri']))
    {
        $message .= 'For the effective URI at ' .
            $result['@metadata']['effectiveUri'] . "\n\n";

        if ((isset($result['Datapoints'])) and
            (count($result['Datapoints']) > 0))
        {
            $message .= "Datapoints found:\n\n";

            foreach($result['Datapoints'] as $datapoint)
            {
                foreach ($datapoint as $key => $value)
                {
                    $message .= $key . ' = ' . $value . "\n";
                }

                $message .= "\n";
            }
        } else {
            $message .= 'No datapoints found.';
        }
    } else {
        $message .= 'No datapoints found.';
    }

    return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
}

function getTheMetricStatistics()
{
    // Average number of Amazon EC2 vCPUs every 5 minutes within
    // the past 3 hours.
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ]
}
```

```
];
$startTime = strtotime('-3 hours');
$endTime = strtotime('now');
$period = 300; // Seconds. (5 minutes = 300 seconds.)
$statistics = array('Average');
$unit = 'None';

$cloudWatchClient = new CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-08-01'
]);

echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);

// Another example: average number of bytes of standard storage in the
// specified Amazon S3 bucket each day for the past 3 days.

/*
$namespace = 'AWS/S3';
$metricName = 'BucketSizeBytes';
$dimensions = [
    [
        'Name' => 'StorageType',
        'Value' => 'StandardStorage'
    ],
    [
        'Name' => 'BucketName',
        'Value' => 'my-bucket'
    ]
];
$startTime = strtotime('-3 days');
$endTime = strtotime('now');
$period = 86400; // Seconds. (1 day = 86400 seconds.)
$statistics = array('Average');
$unit = 'Bytes';

$cloudWatchClient = new CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-08-01'
]);

echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
*/
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

Publicación de métricas personalizadas en Amazon CloudWatch con el AWS SDK for PHP versión 3

Las métricas son los datos sobre el desempeño de los sistemas. Una alarma vigila una métrica individual durante un periodo de tiempo que usted especifica. Realiza una o varias acciones según el valor de la métrica con respecto a un umbral dado durante varios periodos de tiempo.

Los siguientes ejemplos muestran cómo:

- Publicar datos de métricas con [PutMetricData](#).

- Crear una alarma utilizando [PutMetricAlarm](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Publicar datos de métricas

Importaciones

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

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

Código de muestra

```
function putMetricData($cloudWatchClient, $cloudWatchRegion, $namespace,
    $metricData)
{
    try {
        $result = $cloudWatchClient->putMetricData([
            'Namespace' => $namespace,
            'MetricData' => $metricData
        ]);

        if (isset($result['@metadata']['effectiveUri']))
        {
            if ($result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com')
            {
                return 'Successfully published datapoint(s).';
            } else {
                return 'Could not publish datapoint(s).';
            }
        } else {
            return 'Error: Could not publish datapoint(s).';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function putTheMetricData()
{
    $namespace = 'MyNamespace';
    $metricData = [
        [
            'MetricName' => 'MyMetric',
            'Timestamp' => 1589228818, // 11 May 2020, 20:26:58 UTC.
            'Dimensions' => [
                [
                    'Name' => 'MyDimension1',
                    'Value' => 'MyValue1'
                ]
            ],
        ],
    ],
}
```

```
        [
            'Name' => 'MyDimension2',
            'Value' => 'MyValue2'
        ],
        'Unit' => 'Count',
        'Value' => 1
    ]
];

$cloudWatchRegion = 'us-east-1';
$cloudWatchClient = new CloudWatchClient([
    'profile' => 'default',
    'region' => $cloudWatchRegion,
    'version' => '2010-08-01'
]);

echo putMetricData($cloudWatchClient, $cloudWatchRegion, $namespace,
    $metricData);
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricData();
```

Crear una alarma

Importaciones

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

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

Código de muestra

```
function putMetricAlarm($cloudWatchClient, $cloudWatchRegion,
    $alarmName, $namespace, $metricName,
    $dimensions, $statistic, $period, $comparison, $threshold,
    $evaluationPeriods)
{
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri']))
        {
            if ($result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com')
            {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        }
    }
}
```

```
    }
    } else {
        return 'Could not create or update specified alarm.';
    }
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
}

function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm($cloudWatchClient, $cloudWatchRegion,
        $alarmName, $namespace, $metricName,
        $dimensions, $statistic, $period, $comparison, $threshold,
        $evaluationPeriods);
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();
```

Envío de eventos a Amazon CloudWatch Events con el AWS SDK for PHP versión 3

Eventos de CloudWatch proporciona un flujo casi en tiempo real de eventos del sistema que describe cambios en recursos de Amazon Web Services (AWS) en diversos destinos. Mediante reglas sencillas, puede asignar los eventos y dirigirlos a uno o más flujos o funciones de destino.

Los siguientes ejemplos muestran cómo:

- Cree una regla mediante [PutRule](#).
- Añadir destinos a una regla utilizando [PutTargets](#).
- Describe cómo enviar eventos personalizados a Eventos de CloudWatch mediante [PutEvents](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Crear una regla

Importaciones

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

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

Código de muestra

```
$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());
}
```

Añadir destinos a una regla

Importaciones

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

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

Código de muestra

```
$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());
}
```

Enviar eventos personalizados

Importaciones

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

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

Código de muestra

```
$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());
}
```

Uso de acciones de alarma con las alarmas de Amazon CloudWatch con el AWS SDK for PHP versión 3

Use las acciones de alarma para crear alarmas que paran, terminan, reinician o recuperan automáticamente sus instancias Amazon EC2. Puede utilizar las acciones parar o terminar cuando ya no necesita que se ejecute una instancia. Puede usar las acciones reiniciar y recuperar para reiniciar automáticamente esas instancias.

Los siguientes ejemplos muestran cómo:

- Describe cómo habilitar acciones para alarmas especificadas mediante [EnableAlarmActions](#).
- Describe cómo deshabilitar acciones para alarmas especificadas mediante [DisableAlarmActions](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Habilitar acciones de alarma

Importaciones

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

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

Código de muestra

```
function enableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->enableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri']))
        {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been enabled.';
        } else {
            return 'Actions for some matching alarms ' .
                'might not have been enabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function enableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo enableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// enableTheAlarmActions();
```

Deshabilitar acciones de alarma

Importaciones

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

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

Código de muestra

```
function disableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->disableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri']))
        {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been disabled.';
        } else {
            return 'Actions for some matching alarms ' .
                'might not have been disabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function disableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

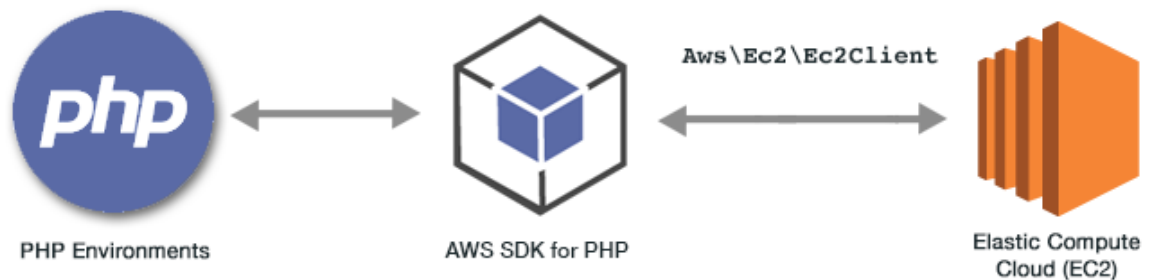
    echo disableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
```

```
// disableTheAlarmActions();
```

Ejemplos de Amazon EC2 que utilizan el AWS SDK for PHP versión 3

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona alojamiento virtual a servidores en la nube. Se ha diseñado para facilitar a los desarrolladores la informática en la nube en la web proporcionándoles capacidad de cómputo variable.



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Temas

- [Administración de instancias Amazon EC2 utilizando el AWS SDK for PHP versión 3](#) (p. 161)
- [Uso de direcciones IP elásticas con Amazon EC2 con el AWS SDK for PHP versión 3](#) (p. 164)
- [Uso de regiones y zonas de disponibilidad para Amazon EC2 con el AWS SDK for PHP versión 3](#) (p. 166)
- [Uso de pares de claves de Amazon EC2 con el AWS SDK for PHP versión 3](#) (p. 167)
- [Uso de grupos de seguridad en Amazon EC2 con el AWS SDK for PHP versión 3](#) (p. 169)

Administración de instancias Amazon EC2 utilizando el AWS SDK for PHP versión 3

Los siguientes ejemplos muestran cómo:

- Describir instancias Amazon EC2 utilizando [DescribeInstances](#).
- Habilite la monitorización detallada de una instancia en ejecución mediante [MonitorInstances](#).
- El monitoreo de una instancia en ejecución se deshabilita mediante [UnmonitorInstances](#).
- Inicie una AMI respaldada por Amazon EBS que haya detenido previamente utilizando [StartInstances](#).
- Detener una instancia respaldada por Amazon EBS utilizando [StopInstances](#).
- Solicite un reinicio de una o varias instancias mediante [RebootInstances](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Describir instancias

Importaciones

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

use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeInstances();

var_dump($result);
```

Habilitar y deshabilitar el monitoreo

Importaciones

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

use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Iniciar y detener una instancia

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Reiniciar una instancia

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Uso de direcciones IP elásticas con Amazon EC2 con el AWS SDK for PHP versión 3

Una dirección IP elástica (EIP) es una dirección IP estática diseñada para la informática en la nube dinámica. Una dirección IP elástica se asocia a su cuenta de AWS. Se trata de una dirección IP pública a la que se puede obtener acceso desde Internet. Si la instancia no tiene una dirección IP pública, puede asociar una dirección IP elástica (EIP) a la instancia para permitir la comunicación con Internet.

Los siguientes ejemplos muestran cómo:

- Describa una o varias de sus instancias mediante [DescribeInstances](#).
- Adquirir una dirección IP elástica utilizando [AllocateAddress](#).
- Asocie una dirección IP elástica a una instancia con [AssociateAddress](#).
- Liberar una dirección IP elástica con [ReleaseAddress](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Describir una instancia

Importaciones

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

use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeInstances();

var_dump($result);
```

Asignar y asociar una dirección

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Liberar una dirección

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Uso de regiones y zonas de disponibilidad para Amazon EC2 con el AWS SDK for PHP versión 3

Amazon EC2 se hospeda en varias ubicaciones de todo el mundo. Dichas ubicaciones se componen de regiones y zonas de disponibilidad de AWS. Cada región es un área geográfica independiente con varias ubicaciones aisladas conocidas como zonas de disponibilidad. Amazon EC2 ofrece la posibilidad de situar instancias y datos en varias ubicaciones.

Los siguientes ejemplos muestran cómo:

- Describa las zonas de disponibilidad que tiene a su disposición mediante [DescribeAvailabilityZones](#).
- Describa las regiones de AWS que están disponibles actualmente mediante [DescribeRegions](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Describir zonas de disponibilidad

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([  
    'region' => 'us-west-2',  
    'version' => '2016-11-15',  
    'profile' => 'default'  
]);  
  
$result = $ec2Client->describeAvailabilityZones();  
  
var_dump($result);
```

Describir regiones

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([  
    'region' => 'us-west-2',  
    'version' => '2016-11-15',  
    'profile' => 'default'  
]);  
  
$result = $ec2Client->describeRegions();  
  
var_dump($result);
```

Uso de pares de claves de Amazon EC2 con el AWS SDK for PHP versión 3

Amazon EC2 utiliza la criptografía de clave pública para cifrar y descifrar la información de inicio de sesión. En la criptografía de clave pública se utiliza una clave pública para cifrar los datos. A continuación, el destinatario utiliza la clave privada para descifrar los datos. El conjunto de clave pública y clave privada se denomina par de claves.

Los siguientes ejemplos muestran cómo:

- Cree un par de claves RSA de 2048 bits utilizando [CreateKeyPair](#).
- Eliminar un par de claves especificado utilizando [DeleteKeyPair](#).
- Describa uno o varios de sus pares de claves mediante [DescribeKeyPairs](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de un par de claves

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Eliminar un par de claves

Importaciones

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

use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Describir pares de claves

Importaciones

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

use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
```

```
$result = $ec2Client->describeKeyPairs();  
  
var_dump($result);
```

Uso de grupos de seguridad en Amazon EC2 con el AWS SDK for PHP versión 3

Un grupo de seguridad de Amazon EC2 funciona como un firewall virtual que controla el tráfico de una o varias instancias. Se añaden reglas a cada grupo de seguridad para permitir el tráfico con sus instancias asociadas. Puede modificar las reglas de un grupo de seguridad en cualquier momento. Las nuevas reglas se aplican automáticamente a todas las instancias asociadas al grupo de seguridad.

Los siguientes ejemplos muestran cómo:

- Describa uno o varios de sus grupos de seguridad mediante [DescribeSecurityGroups](#).
- Añada una regla de entrada a un grupo de seguridad utilizando [AuthorizeSecurityGroupIngress](#).
- Crear un grupo de seguridad utilizando [CreateSecurityGroup](#).
- Eliminar un grupo de seguridad utilizando [DeleteSecurityGroup](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Describir grupos de seguridad

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$ec2Client = new Aws\Ec2\Ec2Client([  
    'region' => 'us-west-2',  
    'version' => '2016-11-15',  
    'profile' => 'default'  
]);  
  
$result = $ec2Client->describeSecurityGroups();  
  
var_dump($result);
```

Añadir una regla de entrada

Importaciones


```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Creación de un grupo de seguridad

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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";
```

Eliminación de un grupo de seguridad

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ec2\Ec2Client;
```

Código de muestra

```
$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);
```

Firma de una petición de búsqueda de Amazon Elasticsearch Service con el AWS SDK for PHP versión 3

Amazon Elasticsearch Service (Amazon ES) es un servicio administrado que facilita las tareas de implementación, operación y escalado de Amazon Elasticsearch Service, un popular motor de búsqueda y análisis de código abierto. Amazon ES ofrece acceso directo a la API de Amazon Elasticsearch Service. Esto significa que los desarrolladores pueden utilizar las herramientas con las que están familiarizados, así como opciones de seguridad sólidas como, por ejemplo, el uso de usuarios y roles de IAM para controlar el acceso. Muchos clientes de Amazon Elasticsearch Service son compatibles con firma de solicitudes, pero si está utilizando un cliente que no lo es, puede firmar solicitudes PSR-7 arbitrarias con los proveedores de credenciales integrados y los firmantes del AWS SDK for PHP.

Los siguientes ejemplos muestran cómo:

- Firme una solicitud con el protocolo de firma de AWS mediante [SignatureV4](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Firma de una solicitud de Amazon ES

Amazon ES usa [Signature Version 4](#). Esto significa que necesita firmar solicitudes para el nombre de la firma del servicio (en este caso es) y la región de AWS de su dominio de Amazon ES. Encontrará una lista completa de regiones con las que Amazon ES es compatible [en la página Regiones y puntos de enlace](#)

de [AWS](#) de la Amazon Web Services General Reference. Sin embargo, en este ejemplo, vamos a firmar solicitudes dirigidas a un dominio de Amazon ES de la región us-west-2.

Tiene que proporcionar credenciales, lo que puede hacer con la cadena de proveedor predeterminada del SDK o con cualquiera de los formatos de credenciales que se describen en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). También necesitará una [solicitud PSR-7](#) (que en el código siguiente se presupone que se llama `$psr7Request`).

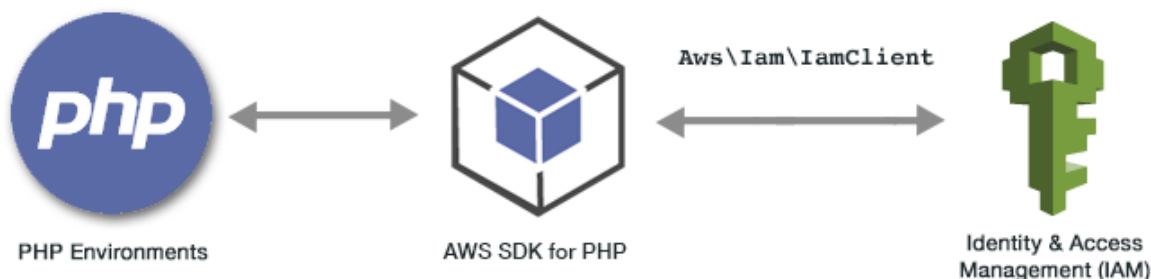
```
// Pull credentials from the default provider chain
$provider = Aws\Credentials\CredentialProvider::defaultProvider();
$credentials = call_user_func($provider)->wait();

// 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);
```

Ejemplos de AWS IAM que utilizan el AWS SDK for PHP versión 3

AWS Identity and Access Management (IAM) es un servicio web que permite a los clientes de Amazon Web Services (AWS) administrar a los usuarios y los permisos de usuario en AWS. El servicio está dirigido a las organizaciones con múltiples usuarios o sistemas en la nube que utilizan productos de AWS. Con IAM puede administrar de forma centralizada usuarios y credenciales de seguridad como las claves de acceso y los permisos que controlan a qué recursos de AWS pueden tener acceso los usuarios.



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Temas

- [Administración de claves de acceso de IAM con el AWS SDK for PHP versión 3](#) (p. 173)
- [Administración de usuarios de IAM con el AWS SDK for PHP versión 3](#) (p. 176)
- [Uso de alias de cuenta de IAM con el AWS SDK for PHP versión 3](#) (p. 179)
- [Uso de políticas de IAM con el AWS SDK for PHP versión 3](#) (p. 181)
- [Uso de certificados de servidor de IAM con el AWS SDK for PHP versión 3](#) (p. 188)

Administración de claves de acceso de IAM con el AWS SDK for PHP versión 3

Los usuarios necesitan sus propias claves de acceso para realizar llamadas por programa a AWS. Para atender esta necesidad, puede crear, modificar, ver o rotar claves de acceso (clave de acceso IDs y claves de acceso secretas) para los usuarios de IAM. De forma predeterminada, cuando se crea una clave de acceso, su estado es activo. Esto significa que el usuario puede utilizar la clave de acceso para realizar llamadas a la API.

Los siguientes ejemplos muestran cómo:

- Cree una clave de acceso secreta y el ID de clave de acceso correspondiente utilizando [CreateAccessKey](#).
- Devuelva información sobre la clave de acceso IDs asociada a un usuario de IAM mediante [ListAccessKeys](#).
- Recupere información acerca de cuándo se utilizó por última vez una clave de acceso con [GetAccessKeyLastUsed](#).
- Cambie el estado de una clave de acceso de activa a inactiva o viceversa utilizando [UpdateAccessKey](#).
- Elimine un par de claves de acceso asociado a un usuario de IAM mediante [DeleteAccessKey](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Crear una clave de acceso

Importaciones

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

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

Código de muestra

```
$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());
}
```

Enumerar las claves de acceso

Importaciones

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

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

Código de muestra

```
$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());
}
```

Obtener más información el último uso de una clave de acceso

Importaciones

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

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

Código de muestra

```
$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());
}
```

Actualizar una clave de acceso

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminar una clave de acceso

Importaciones

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

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

Código de muestra

```
$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());
}
```

Administración de usuarios de IAM con el AWS SDK for PHP versión 3

Un usuario de IAM es una entidad que se crea en AWS para representar a la persona o al servicio que lo utiliza para interactuar con AWS. Un usuario de AWS consta de un nombre y de credenciales.

Los siguientes ejemplos muestran cómo:

- Cree un nuevo usuario de IAM mediante [CreateUser](#).
- Muestre los usuarios de IAM que usan [ListUsers](#).
- Actualice un usuario de IAM con [UpdateUser](#).
- Recupere información acerca de un usuario de IAM mediante [GetUser](#).
- Elimine un usuario de IAM mediante [DeleteUser](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de un usuario de IAM

Importaciones

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

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

Código de muestra

```
$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());
}
```

Listar usuarios de IAM

Importaciones

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

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

Código de muestra

```
$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());
}
```

Actualizar un usuario de IAM

Importaciones

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

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

Código de muestra

```
$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());
}
```

Obtener información acerca de un usuario de IAM

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminar un usuario de IAM

Importaciones

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

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

Código de muestra

```
$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());  
}
```

Uso de alias de cuenta de IAM con el AWS SDK for PHP versión 3

Si quiere que la dirección URL de la página de inicio de sesión contenga el nombre de su empresa u otro identificador intuitivo en lugar de su ID de cuenta de AWS, puede crear un alias para el ID de cuenta de AWS. Si crea un alias de cuenta de AWS, la dirección URL de su página de inicio de sesión cambia para incorporar dicho alias.

Los siguientes ejemplos muestran cómo:

- Crear un alias utilizando [CreateAccountAlias](#).
- Enumere el alias asociado a la cuenta de AWS mediante [ListAccountAliases](#).
- Eliminar un alias utilizando [DeleteAccountAlias](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de un alias

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Iam\IamClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Enumerar alias de cuenta

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminar un alias

Importaciones

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

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

Código de muestra

```
$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());
}
```

Uso de políticas de IAM con el AWS SDK for PHP versión 3

Los permisos a un usuario se conceden mediante la creación de una política. Una política es un documento que incluye una lista de las acciones que puede realizar un usuario y los recursos a los que pueden afectar dichas acciones. De forma predeterminada, todas las acciones o recursos que no se permiten de forma explícita se rechazan. Puede crear políticas y asociarlas a usuarios, a grupos de usuarios, a roles asumidos por usuarios y a recursos.

Los siguientes ejemplos muestran cómo:

- Crear una política administrada utilizando [CreatePolicy](#).
- Asocie una política a un rol utilizando [AttachRolePolicy](#).
- Asocie una política a un usuario mediante [AttachUserPolicy](#).
- Asociar una política a un grupo utilizando [AttachGroupPolicy](#).
- Quitar una política de rol utilizando [DetachRolePolicy](#).
- Quitar una política de usuario utilizando [DetachUserPolicy](#).
- Quitar una política de grupo utilizando [DetachGroupPolicy](#).
- Elimine una política administrada mediante [DeletePolicy](#).
- Describe cómo eliminar una política de rol mediante [DeleteRolePolicy](#).
- Eliminar una política de usuario utilizando [DeleteUserPolicy](#).
- Elimine una política de grupo mediante [DeleteGroupPolicy](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de una política

Importaciones

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

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

Código de muestra

```
$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());
}
```

Asociación de una política a un rol

Importaciones

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

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

Código de muestra

```
$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());
}
```

Asociación de una política a un usuario

Importaciones

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

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

Código de muestra

```
$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());
}
```

Asociación de una política a un grupo

Importaciones

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

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

Código de muestra

```
$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());
}
```

Desasociación de una política de usuario

Importaciones

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

use Aws\Iam\IamClient;
```

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

Código de muestra

```
$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());
}
```

Desasociación de una política de grupo

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminación de una política

Importaciones


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

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

Código de muestra

```
$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());
}
```

Eliminación de una política de rol

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminación de una política de usuario

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminación de una política de grupo

Importaciones

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

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

Código de muestra

```
$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());
}
```

Uso de certificados de servidor de IAM con el AWS SDK for PHP versión 3

Para habilitar las conexiones HTTPS en su sitio web o aplicación en AWS, necesita un certificado de servidor de SSL/TLS. Para utilizar un certificado obtenido de un proveedor externo en su sitio web o aplicación en AWS, debe cargar el certificado en IAM o importarlo a AWS Certificate Manager.

Los siguientes ejemplos muestran cómo:

- Muestre los certificados almacenados en IAM mediante [ListServerCertificates](#).
- Recupere información acerca de un certificado mediante [GetServerCertificate](#).
- Actualizar un certificado mediante [UpdateServerCertificate](#).
- Elimine un certificado con [DeleteServerCertificate](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Crear una lista de certificados de servidor

Importaciones

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

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

Código de muestra

```
$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());
}
```

Recuperar un certificado de servidor

Importaciones

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

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

Código de muestra

```
$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());
}
```

Actualizar un certificado de servidor

Importaciones

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

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

Código de muestra

```
$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());
}
```

Eliminar un certificado de servidor

Importaciones

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

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

Código de muestra

```
$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());
}
```

Ejemplos de AWS Key Management Service que utilizan el AWS SDK for PHP versión 3

AWS Key Management Service (AWS KMS) es un servicio administrado que le permite crear y controlar fácilmente las claves de cifrado que se utilizan para cifrar datos. Para obtener más información acerca de AWS KMS, consulte la [documentación de Amazon KMS](#). Tanto si escribe aplicaciones seguras para PHP como si envía datos a otros servicios de AWS, AWS KMS le ayuda a mantener el control sobre quién puede usar las claves maestras y obtener acceso a sus datos cifrados.



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- [Uso de claves mediante la API de AWS KMS y el AWS SDK for PHP versión 3](#) (p. 191)
- [Cifrado y descifrado de claves de datos de AWS KMS mediante el AWS SDK for PHP versión 3](#) (p. 195)
- [Uso de las políticas de claves de AWS KMS con el AWS SDK for PHP versión 3](#) (p. 197)
- [Uso de concesiones mediante la API de AWS KMS y el AWS SDK for PHP versión 3](#) (p. 200)
- [Uso de alias mediante la API de AWS KMS y el AWS SDK for PHP versión 3](#) (p. 204)

Uso de claves mediante la API de AWS KMS y el AWS SDK for PHP versión 3

Los recursos principales de AWS Key Management Service (AWS KMS) son las [claves maestras de cliente \(CMK\)](#). Puede usar CMK para cifrar sus datos.

Los siguientes ejemplos muestran cómo:

- Crear una CMK de cliente mediante [CreateKey](#).
- Generar una clave de datos utilizando [GenerateDataKey](#).
- Describe cómo ver una CMK mediante [DescribeKey](#).
- Obtenga la clave IDs y los ARN de clave de CMKs mediante [ListKeys](#).
- Habilite CMKs mediante [EnableKey](#).
- Deshabilite CMKs mediante [DisableKey](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Para obtener más información acerca de AWS Key Management Service (AWS KMS), consulte [Guía para desarrolladores de AWS KMS](#).

Cree una clave CMK

Para crear una [CMK](#), utilice la operación [CreateKey](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Generar una clave de datos

Para generar una clave de cifrado de datos, utilice la operación [GenerateDataKey](#). Esta operación devuelve copias en texto no cifrado y cifradas de la clave de datos que crea. Especifique la clave maestra de cliente (CMK) con la que se va a generar la clave de datos.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Kms\KmsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Visualización de una CMK

Para obtener información detallada sobre una CMK, incluido el nombre de recurso de Amazon (ARN) de la CMK y el [estado de clave](#), utilice la operación [DescribeKey](#).

DescribeKey no obtiene los alias. Para obtener los alias, utilice la operación [ListAliases](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Obtenga el ID de la clave y la clave ARNs de una CMK

Para obtener el ID y el ARN de la CMK, utilice la operación [ListAliases](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Habilitación de una CMK

Para habilitar una CMK deshabilitada, utilice la operación [EnableKey](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Kms\KmsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Deshabilitación de una CMK

Para deshabilitar una CMK, utilice la operación [DisableKey](#). Al deshabilitar una CMK se impide que se utilice.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Kms\KmsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Cifrado y descifrado de claves de datos de AWS KMS mediante el AWS SDK for PHP versión 3

Las claves de datos son las claves de cifrado que puede utilizar para cifrar los datos, incluidas grandes cantidades de datos y otras claves de cifrado de datos.

Puede utilizar una [clave maestra del cliente \(CMK\)](#) de AWS Key Management Service (AWS KMS) para generar, cifrar y descifrar claves de datos. Sin embargo, AWS KMS no almacena, administra o realiza el seguimiento de las claves de datos, ni realiza operaciones criptográficas con claves de datos. Las claves de datos se utilizan y administran fuera de AWS KMS.

Los siguientes ejemplos muestran cómo:

- Cifrar una clave de datos mediante [Encrypt](#).
- Descifrar una clave de datos mediante [Decrypt](#).
- Volver a cifrar una clave de datos con una nueva CMK con [ReEncrypt](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca de AWS Key Management Service (AWS KMS), consulte [Guía para desarrolladores de AWS KMS](#).

Encrypt

La operación [Encrypt](#) se ha diseñado para cifrar claves de datos, pero no se utiliza con frecuencia. Las operaciones [GenerateDataKey](#) y [GenerateDataKeyWithoutPlaintext](#) devuelven claves de datos cifradas. Podría utilizar el método `Encrypt` cuando mueva datos cifrados a una nueva región de AWS y desee cifrar su clave de datos con una CMK en la nueva región.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$message = pack('c*', 1, 2, 3, 4, 5, 6, 7, 8, 9, 0);

try {
    $result = $KmsClient->encrypt([
        'KeyId' => $keyId,
        'Plaintext' => $message,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Decrypt

Para descifrar una clave de datos, use la operación [Decrypt](#).

El `ciphertextBlob` que especifique debe ser el valor del campo `CiphertextBlob` de una respuesta [GenerateDataKey](#), [GenerateDataKeyWithoutPlaintext](#) o [Encrypt](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Reencrypt

Para descifrar una clave de datos cifrada y, a continuación, volver a cifrar inmediatamente la clave de datos con una CMK diferente, utilice la operación [ReEncrypt](#). Las operaciones se realizan en su totalidad en el servidor en AWS KMS, por lo que su texto no cifrado nunca se expondrá fuera de AWS KMS.

El `ciphertextBlob` que especifique debe ser el valor del campo `CiphertextBlob` de una respuesta [GenerateDataKey](#), [GenerateDataKeyWithoutPlaintext](#) o [Encrypt](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Kms\KmsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Uso de las políticas de claves de AWS KMS con el AWS SDK for PHP versión 3

Al crear una [clave maestra de cliente \(CMK\)](#) de AWS Key Management Service (AWS KMS), determine quién puede usar y administrar dicha CMK. Estos permisos se encuentran en un documento denominado

la política de claves. Puede utilizar la política de claves para agregar, eliminar o modificar permisos en cualquier momento para una CMK administrada por el cliente, pero no puede editar la política de claves para una CMK administrada por AWS. Para obtener más información, consulte [Autenticación y control de acceso de AWS KMS](#).

Los siguientes ejemplos muestran cómo:

- Enumere los nombres de las políticas de claves mediante [ListKeyPolicies](#).
- Obtenga una política de claves utilizando [GetKeyPolicy](#).
- Establezca una política de claves mediante [PutKeyPolicy](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca de AWS Key Management Service (AWS KMS), consulte [Guía para desarrolladores de AWS KMS](#).

Enumerar todas las políticas de claves

Para obtener los nombres de las políticas de claves de una CMK, utilice la operación `ListKeyPolicies`. El único nombre de política de claves que se devuelve es el nombre predeterminado.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Recuperar una política de claves

Para obtener la política de clave de una CMK, utilice la operación `GetKeyPolicy`.

`GetKeyPolicy` requiere un nombre de política. A menos que haya creado una política de claves al crear la CMK, el único nombre de política válido es el predeterminado. Obtenga más información sobre la [política de claves predeterminada](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Establecer una política de claves

Para establecer o cambiar una política de claves para una CMK, utilice la operación `PutKeyPolicy`.

`PutKeyPolicy` requiere un nombre de política. A menos que haya creado una política de claves al crear la CMK, el único nombre de política válido es el predeterminado. Obtenga más información sobre la [política de claves predeterminada](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Uso de concesiones mediante la API de AWS KMS y el AWS SDK for PHP versión 3

Un concesión es otro mecanismo para proporcionar permisos, una alternativa a la política de claves. Puede utilizar las concesiones para ofrecer acceso a largo plazo que permita a las entidades principales de AWS utilizar las AWS Key Management Service administradas por el cliente AWS KMS de (CMKs). Para obtener más información, consulte [Uso de concesiones](#).

Los siguientes ejemplos muestran cómo:

- Creación de una concesión para una clave maestra de cliente (CMK) con [CreateGrant](#).
- Visualización de una concesión para una CMK con [ListGrants](#).

- Retire una concesión para una CMK con [RetireGrant](#).
- Revoque una concesión para una CMK con [RevokeGrant](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca de AWS Key Management Service (AWS KMS), consulte [Guía para desarrolladores de AWS KMS](#).

Crear una concesión

Para crear una concesión para una CMK de AWS KMS, utilice la operación [CreateGrant](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Visualización de una concesión

Para obtener información detallada sobre las concesiones de una CMK de AWS KMS, utilice la operación [ListGrants](#).

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Retirada de una concesión

Para retirar una concesión para una CMK de AWS KMS, utilice la operación [RetireGrant](#). Retire una concesión para efectuar una limpieza cuando termine de utilizarla.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Revocación de una concesión

Para revocar una concesión a una CMK de AWS KMS, utilice la operación [RevokeGrant](#). Puede revocar una concesión para denegar explícitamente las operaciones que dependen de ella.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Uso de alias mediante la API de AWS KMS y el AWS SDK for PHP versión 3

Un alias es un nombre de visualización opcional para una [clave maestra de cliente \(CMK\)](#) de AWS Key Management Service (AWS KMS).

Los siguientes ejemplos muestran cómo:

- Crear un alias utilizando [CreateAlias](#).
- Ver un alias mediante [ListAliases](#).
- Actualizar un alias mediante [UpdateAlias](#).
- Eliminar un alias utilizando [DeleteAlias](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca de AWS Key Management Service (AWS KMS), consulte [Guía para desarrolladores de AWS KMS](#).

Creación de un alias

Para crear un alias para una CMK, utilice la operación [CreateAlias](#). El alias debe ser único en la cuenta y región de AWS. Si crea un alias para una CMK que ya tiene un alias, [CreateAlias](#) crea otro alias para la misma CMK. No sustituye el alias existente.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Ver un alias

Para enumerar todos los alias, utilice la operación [ListAliases](#). La respuesta incluye los alias definidos por los servicios de AWS, pero no están asociados a una CMK.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Actualizar un alias

Para asociar un alias existente con una CMK diferente, utilice la operación [UpdateAlias](#).

Importaciones

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

use Aws\Kms\KmsClient;
```

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

Código de muestra

```
$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";
}
```

Eliminar un alias

Para eliminar un alias, utilice la operación [DeleteAlias](#). La eliminación de un alias no afecta a la CMK subyacente.

Importaciones

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

use Aws\Kms\KmsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Ejemplos de Amazon Kinesis que utilizan el AWS SDK for PHP versión 3

Amazon Kinesis es un servicio de AWS que recopila, procesa y analiza datos en tiempo real. Configure las secuencias de datos con Amazon Kinesis Data Streams o utilice Amazon Kinesis Data Firehose para enviar datos a Amazon S3, Amazon ES, Amazon Redshift o Splunk.

Para obtener más información acerca de Kinesis, consulte la [documentación de Amazon Kinesis](#).



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- [Creación de secuencias de datos mediante la API de Kinesis Data Streams y AWS SDK for PHP versión 3](#) (p. 207)
- [Administración de fragmentos de datos mediante la API de Kinesis Data Streams y AWS SDK for PHP versión 3](#) (p. 211)
- [Creación de secuencias de entrega mediante la API de Kinesis Data Firehose y AWS SDK for PHP versión 3](#) (p. 213)

Creación de secuencias de datos mediante la API de Kinesis Data Streams y AWS SDK for PHP versión 3

Amazon Kinesis Data Streams permite enviar datos en tiempo real. Cree un productor de datos con Kinesis Data Streams que entregue datos al destino configurado cada vez que agregue datos.

Para obtener más información, consulte [Creación y administración de secuencias](#) en la Amazon Kinesis Data Streams Developer Guide.

Los siguientes ejemplos muestran cómo:

- Describe cómo crear una secuencia de datos con [CreateAlias](#).
- Obtenga información detallada acerca de una única secuencia de datos con [DescribeStream](#).
- Muestre las secuencias de datos existentes con [ListStreams](#).
- Describe cómo enviar datos a una secuencia de datos existente mediante [PutRecord](#).
- Elimine una secuencia de datos con [DeleteStream](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon Kinesis Data Streams, consulte la [Guía para desarrolladores de Amazon Kinesis Data Streams](#).

Creación de una secuencia de datos mediante una secuencia de datos de Kinesis

Establezca una secuencia de datos de Kinesis en la que pueda enviar información que procesará Kinesis mediante el siguiente ejemplo de código. Obtenga más información sobre [cómo crear y actualizar secuencias de datos](#) en Amazon Kinesis Data Streams Developer Guide.

Para crear una secuencia de datos de Kinesis, utilice la operación [CreateStream](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Recuperación de una secuencia de datos

Obtenga información detallada acerca de una secuencia de datos existente mediante el siguiente ejemplo de código. De forma predeterminada, este devuelve información acerca de los primeros 10 fragmentos conectados a la secuencia de datos de Kinesis especificada. Recuerde comprobar `StreamStatus` desde la respuesta antes de escribir datos en una secuencia de datos de Kinesis.

Para recuperar detalles sobre una secuencia de datos de Kinesis especificada, utilice la operación [DescribeStream](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Lista de secuencias de datos existentes que están conectadas a Kinesis

Enumere las primeras 10 secuencias de datos desde su cuenta de AWS en la región de AWS seleccionada. Utilice el `HasMoreStreams` devuelto para determinar si hay más secuencias asociadas a su cuenta.

Para obtener una lista de las secuencias de datos de Kinesis, utilice la operación [ListStreams](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Envío de datos a una secuencia de datos existente

Una vez que haya creado una secuencia de datos, utilice el siguiente ejemplo para enviar datos. Antes de enviarle los datos, utilice `DescribeStream` para comprobar si los datos `StreamStatus` están activos.

Para escribir un solo registro de datos en una secuencia de datos de Kinesis, utilice la operación [PutRecord](#). Para escribir hasta 500 registros en una secuencia de datos de Kinesis, utilice la operación [PutRecords](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Eliminación de una secuencia de datos

En este ejemplo se muestra cómo se elimina una secuencia de datos. Eliminar una secuencia de datos, también elimina todos los datos que haya enviado a la secuencia de datos. Las secuencias de datos de Kinesis activas cambian al estado `ELIMINANDO` hasta que se haya completado la secuencia de eliminación. Mientras se encuentra en el estado `ELIMINANDO`, la secuencia sigue procesando datos.

Para eliminar una secuencia de datos de Kinesis, utilice la operación [DeleteStream](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Administración de fragmentos de datos mediante la API de Kinesis Data Streams y AWS SDK for PHP versión 3

Amazon Kinesis Data Streams permite enviar datos en tiempo real a un punto de enlace. La velocidad del flujo de datos depende del número de fragmentos de la secuencia.

Puede escribir 1000 registros por segundo en un único fragmento. Cada fragmento también tiene un límite de carga de 1 MiB por segundo. El uso se calcula y se cobra por fragmento, así que debe usar estos ejemplos para administrar la capacidad de los datos y el costo de la secuencia.

Los siguientes ejemplos muestran cómo:

- Muestre los fragmentos de una secuencia mediante [ListShards](#).
- Añada o reduzca el número de fragmentos de una secuencia mediante [UpdateShardCount](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon Kinesis Data Streams, consulte la [Guía para desarrolladores de Amazon Kinesis Data Streams](#).

Lista de fragmentos de secuencias de datos

Enumere los detalles de hasta 100 fragmentos de una secuencia específica.

Para enumerar los fragmentos de una secuencia de datos de Kinesis, utilice la operación [ListShards](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Añadir más fragmentos de secuencias de datos

Si necesita más fragmentos de secuencias de datos, puede aumentar su número actual de fragmentos. Le recomendamos que duplique el recuento de fragmentos cuando aumente. Esto crea una copia de cada partición disponible actualmente para aumentar la capacidad. Puede duplicar el número de los fragmentos solo dos veces en un periodo de 24 horas.

Recuerde que la facturación por el uso de Kinesis Data Streams se calcula por partición, de manera que, cuando la demanda disminuye, le recomendamos que reduzca su recuento de fragmentos a la mitad. Al eliminar los fragmentos, solo puede reducir la cantidad de fragmentos a la mitad de su recuento de fragmentos actual.

Para actualizar el recuento de fragmentos de una secuencia de datos de Kinesis, utilice la operación [UpdateShardCount](#).

Importaciones

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

use Aws\Kinesis\KinesisClient;
```

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

Código de muestra

```
$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";
}
```

Creación de secuencias de entrega mediante la API de Kinesis Data Firehose y AWS SDK for PHP versión 3

Amazon Kinesis Data Firehose le permite enviar datos en tiempo real a otros servicios de AWS, incluidos Amazon Kinesis Data Streams, Amazon S3, Amazon Elasticsearch Service (Amazon ES) y Amazon Redshift, o bien a Splunk. Cree un productor de datos con secuencias de entrega que entregue datos al destino configurado cada vez que agregue datos.

Los siguientes ejemplos muestran cómo:

- Describe cómo crear una secuencia de entrega de con [CreateDeliveryStream](#).
- Obtenga información detallada acerca de un único flujo de entrega con [DescribeDeliveryStream](#).
- Obtenga una lista de las secuencias de entrega con [ListDeliveryStreams](#).
- Describe cómo enviar datos a una secuencia de entrega de mediante [PutRecord](#).
- Elimine una secuencia de entrega de con [DeleteDeliveryStream](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon Kinesis Data Firehose, consulte la [Guía para desarrolladores de Amazon Kinesis Data Firehose](#).

Creación de una secuencia de entrega con una secuencia de datos de Kinesis

Para establecer una secuencia de entrega que coloque datos en una secuencia de datos de Kinesis existente, utilice la operación [CreateDeliveryStream](#).

Esto permite a los desarrolladores migrar los servicios de Kinesis existentes a Kinesis Data Firehose.

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Creación de una secuencia de entrega con buckets de Amazon S3

Para establecer una secuencia de entrega que coloque los datos en un bucket de Amazon S3 existente, utilice la operación [CreateDeliveryStream](#).

Proporcione los parámetros del destino, tal y como se describe en [Parámetros de destino](#). A continuación, asegúrese de conceder a Kinesis Data Firehose acceso a su bucket de Amazon S3, tal y como se describe en la sección sobre cómo [conceder a Kinesis Data Firehose acceso a un destino de Amazon S3](#).

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Creación de una secuencia de entrega con Amazon ES

Para establecer una secuencia de entrega de Kinesis Data Firehose que coloque los datos en un clúster de Amazon ES, utilice la operación [CreateDeliveryStream](#).

Proporcione los parámetros del destino, tal y como se describe en [Parámetros de destino](#). Asegúrese de conceder a Kinesis Data Firehose acceso a su clúster de Amazon ES, tal y como se describe en la sección sobre cómo [conceder a Kinesis Data Firehose acceso a un destino de Amazon ES](#).

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Recuperación de una secuencia de entrega

Para obtener información detallada sobre un flujo de entrega de Kinesis Data Firehose existente, utilice la operación [DescribeDeliveryStream](#).

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Enumeración de secuencias de entrega existentes que están conectadas a Kinesis Data Streams

Para enumerar todas las secuencias de entrega de Kinesis Data Firehose existentes que envían datos a Kinesis Data Streams, utilice la operación [ListDeliveryStreams](#).

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Enumeración de secuencias de entrega existentes que envían datos a otros servicios de AWS

Para enumerar todas las secuencias de entrega de Kinesis Data Firehose existentes que envían datos a Amazon S3, Amazon ES o Amazon Redshift, o bien a Splunk, utilice la operación [ListDeliveryStreams](#).

Importaciones


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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Envío de datos a una secuencia de entrega de Kinesis Data Firehose

Para enviar datos a través de una secuencia de entrega de Kinesis Data Firehose a su destino especificado, utilice la operación [PutRecord](#) después de crear una secuencia de entrega de Kinesis Data Firehose.

Antes de enviar datos a una secuencia de entrega de Kinesis Data Firehose, utilice `DescribeDeliveryStream` para ver si la secuencia de entrega está activa.

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Eliminación de una secuencia de entrega de Kinesis Data Firehose

Para eliminar una secuencia de entrega de Kinesis Data Firehose, utilice la operación [DeleteDeliveryStreams](#). Esta operación también elimina todos los datos que haya enviado a la secuencia de entrega.

Importaciones

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

use Aws\Firehose\FirehoseClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

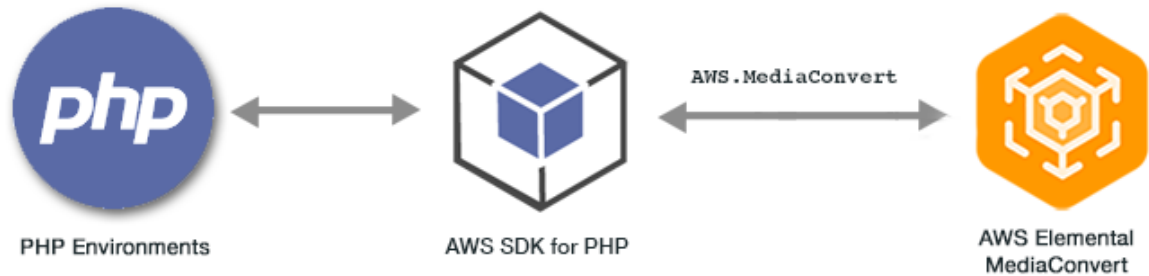
$name = "my_stream_name";

try {
    $result = $firehoseClient->deleteDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Ejemplos de AWS Elemental MediaConvert que utilizan el AWS SDK for PHP versión 3

AWS Elemental MediaConvert es un servicio de transcodificación de vídeo basado en archivos con características apropiadas para los medios de difusión. Puede utilizarlo para crear recursos para

su difusión y para la entrega de vídeo bajo demanda (VOD) a través de Internet. Para obtener más información, consulte la [Guía del usuario de AWS Elemental MediaConvert](#).



La API de PHP para AWS Elemental MediaConvert se expone a través de la clase de cliente `AWS.MediaConvert`. Para obtener más información, consulte [Clase: `AWS.MediaConvert`](#) en la referencia de la API.

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Temas

- [Obtención del punto de enlace específico de la cuenta para AWS Elemental MediaConvert con el AWS SDK for PHP versión 3](#) (p. 220)
- [Creación y administración de trabajos de transcodificación en AWS Elemental MediaConvert con el AWS SDK for PHP versión 3](#) (p. 222)

Obtención del punto de enlace específico de la cuenta para AWS Elemental MediaConvert con el AWS SDK for PHP versión 3

En este ejemplo, se utiliza el AWS SDK for PHP versión 3 para llamar a AWS Elemental MediaConvert y recuperar el punto de enlace específico de la cuenta. Puede recuperar la URL del punto de enlace desde el punto de enlace predeterminado del servicio, por lo que todavía no necesita el punto de enlace específico de la cuenta.

Los siguientes ejemplos muestran cómo:

- Recupere su punto de enlace específico de la cuenta. usando [DescribeEndpoints](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Para tener acceso al cliente de MediaConvert, cree un rol de IAM que conceda a AWS Elemental MediaConvert acceso a los archivos de entrada y a los buckets de Amazon S3 donde se almacenan los archivos de salida. Para obtener más información, consulte [Configuración de permisos de IAM](#) en la [Guía del usuario de AWS Elemental MediaConvert](#).

Recuperación de los puntos de enlace

Cree un objeto para pasar los parámetros de solicitud vacíos para el método `describeEndpoints` de la clase de cliente `Aws\MediaConvert\MediaConvertClient`. Para llamar al método `describeEndpoints`, cree una promesa para invocar un objeto de servicio de AWS Elemental MediaConvert mediante los parámetros. Gestione la respuesta en la devolución de llamada de la promesa.

Importaciones

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

use Aws\MediaConvert\MediaConvertClient;
use Aws\Exception\AwsException;
```

Código de muestra

Defina la región en la que se va a obtener el punto de enlace y cree un objeto de cliente de MediaConvert:

```
$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";
}
```

Llame al método `describeEndpoints` para recuperar los puntos de enlace y guardar la URL del punto de enlace:

```
$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
]);
```

Creación y administración de trabajos de transcodificación en AWS Elemental MediaConvert con el AWS SDK for PHP versión 3

En este ejemplo, se utiliza el AWS SDK for PHP versión 3 para llamar a AWS Elemental MediaConvert y crear un trabajo de transcodificación. Antes de comenzar, tiene que cargar el vídeo de entrada en el bucket de Amazon S3 que ha aprovisionado para el almacenamiento de entrada. Para obtener una lista de los códecs y contenedores de vídeo de entrada admitidos, consulte [Códex y contenedores de entrada admitidos](#) en la [Guía del usuario de AWS Elemental MediaConvert](#).

Los siguientes ejemplos muestran cómo:

- Crear trabajos de transcodificación en AWS Elemental MediaConvert. : [CreateJob](#).
- Cancelar un trabajo de transcodificación de la cola de AWS Elemental MediaConvert. [CancelJob](#)
- Recuperar el JSON de un trabajo de transcodificación completado. [GetJob](#)
- Recuperar una matriz JSON para un máximo de 20 de los últimos trabajos creados. [ListJobs](#)

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para tener acceso al cliente de MediaConvert, cree un rol de IAM que conceda a AWS Elemental MediaConvert acceso a los archivos de entrada y a los buckets de Amazon S3 donde se almacenan los archivos de salida. Para obtener más información, consulte [Configuración de permisos de IAM](#) en la [Guía del usuario de AWS Elemental MediaConvert](#).

Creación de un cliente

Configure el AWS SDK for PHP mediante la creación de un cliente de MediaConvert, con la región para el código. En este ejemplo, la región se establece en us-west-2. Dado que AWS Elemental MediaConvert utiliza puntos de enlace personalizados para cada cuenta, configure la `Aws\MediaConvert\MediaConvertClient` class para que utilice el punto de enlace específico de su cuenta. Para ello, establezca el parámetro de punto de enlace en el [punto de enlace específico de su cuenta \(p. 220\)](#).

Importaciones

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

use Aws\MediaConvert\MediaConvertClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default',
```

```
'endpoint' => 'ACCOUNT_ENDPOINT'  
]);
```

Definición de un trabajo de transcodificación sencillo

Cree el JSON que define los parámetros del trabajo de transcodificación.

Estos parámetros son detallados. Puede utilizar la [consola de MediaConvert de AWS Elemental](#) para generar los parámetros del trabajo JSON eligiendo la configuración del trabajo en la consola y, a continuación, eligiendo Show job JSON (Mostrar JSON del trabajo) en la parte inferior de la sección Job (Trabajo). En este ejemplo, se muestra el JSON de un trabajo sencillo.

Código de muestra

```
$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"
    ]
]
```

Creación de un trabajo

Después de crear el JSON de parámetros de trabajo, llame al método `createJob` invocando un `AWS.MediaConvert service object` y pasando los parámetros. El ID del trabajo creado se devuelve en los datos de la respuesta.

Código de muestra

```
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";
}
```

Recuperación de un trabajo

Con el `JobID` devuelto cuando llamó a `createjob`, puede obtener descripciones detalladas de trabajos recientes en formato JSON.

Código de muestra

```
$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";
}
```


Cancelación de un trabajo

Con el JobID que se devuelve al llamar a createjob, se puede cancelar un trabajo mientras esté en la cola. No se pueden cancelar los trabajos cuya transcodificación ya ha comenzado.

Código de muestra

```
$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";
}
```

Listado de los trabajos de transcodificación recientes

Cree el JSON con los parámetros, incluidos los valores que especifican si se debe ordenar la lista en orden ascendente (ASCENDING) o descendente (DESCENDING), el ARN de la cola de trabajos que se va comprobar y el estado de los trabajos que se deben incluir. Esto devuelve hasta 20 trabajos. Para recuperar los siguientes 20 trabajos más recientes, utilice la cadena nextToken devuelta con el resultado.

Código de muestra

```
$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";
}
```

Ejemplos de Amazon S3 que utilizan el AWS SDK for PHP versión 3

Amazon Simple Storage Service (Amazon S3) es un servicio web que ofrece un sistema de almacenamiento en la nube altamente escalable. Amazon S3 proporciona un almacenamiento de objetos fácil de usar, con una interfaz de servicios web sencilla que puede utilizarse para almacenar y recuperar cualquier cantidad de datos desde cualquier lugar de la web.



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Temas

- [Creación y uso de buckets de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 227\)](#)
- [Administración de permisos de acceso a los buckets de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 230\)](#)
- [Configuración de buckets de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 231\)](#)
- [Describe cómo utilizar cargas multiparte de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 233\)](#)
- [Amazon S3 prefirmada de POSTs con el AWS SDK for PHP versión 3 \(p. 240\)](#)
- [URL prefirmada de Amazon S3 con el AWS SDK for PHP versión 3 \(p. 241\)](#)
- [Uso de un bucket de Amazon S3 como un host web estático con el AWS SDK for PHP versión 3 \(p. 243\)](#)
- [Uso de políticas de bucket de Amazon S3 con el AWS SDK para PHP versión 3 \(p. 244\)](#)
- [Con el punto de acceso de S3 ARNs el AWS SDK para PHP versión 3 \(p. 246\)](#)

Creación y uso de buckets de Amazon S3 con el AWS SDK para PHP versión 3

Los siguientes ejemplos muestran cómo:

- : devuelve una lista de buckets pertenecientes al remitente autenticado de la solicitud usando [ListBuckets](#).
- Cree un nuevo bucket con [CreateBucket](#).

- Añada un objeto a un bucket con [PutObject](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Importaciones

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

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

Lista de buckets

Cree un archivo PHP con el siguiente código. En primer lugar, cree un servicio cliente AWS.S3 que especifique la región de AWS y la versión. A continuación, llame al método `listBuckets`, que devuelve todos los buckets de Amazon S3 que pertenecen al remitente de la solicitud como una matriz de estructuras Bucket.

Código de muestra

```
$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";
}
```

Crear un bucket

Cree un archivo PHP con el siguiente código. En primer lugar, cree un servicio cliente AWS.S3 que especifique la región de AWS y la versión. Llame entonces al método `createBucket` con una matriz como parámetro. El único campo obligatorio es la clave 'Bucket', con un valor de cadena para el nombre del bucket que desea crear. Sin embargo, puede especificar también la región de AWS con el campo "CreateBucketConfiguration". Si se ejecuta correctamente, este método devuelve la ubicación del bucket.

Código de muestra

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
            'Bucket' => $bucketName,
        ]);
        return 'The bucket\'s location is: ' .
    }
```

```
        $result['Location'] . ' ' . ' ' .  
        'The bucket\'s effective URI is: ' .  
        $result['@metadata']['effectiveUri'];  
    } catch (AwsException $e) {  
        return 'Error: ' . $e->getAwsErrorMessage();  
    }  
}  
  
function createTheBucket()  
{  
    $s3Client = new S3Client([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => '2006-03-01'  
    ]);  
  
    echo createBucket($s3Client, 'my-bucket');  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// createTheBucket();
```

Colocar un objeto en un bucket

Para añadir archivos a su nuevo bucket, cree un archivo PHP con el código siguiente.

Ejecute este archivo en la línea de comandos y especificando una cadena con el nombre del bucket en el que desea cargar su archivo, seguido de la ruta completa del archivo que desea cargar.

Código de muestra

```
$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";  
}
```

Administración de permisos de acceso a los buckets de Amazon S3 con el AWS SDK para PHP versión 3

Las listas de control de acceso (ACL) son una de las opciones de política de acceso basada en recursos que puede utilizar para administrar el acceso a sus buckets y objetos. Puede utilizar ACLs para conceder permisos de lectura/escritura básicos a otras cuentas de AWS. Para obtener más información, consulte [Administración de acceso con ACLs](#).

El siguiente ejemplo muestra cómo:

- Obtener la política de control de acceso para un bucket utilizando [GetBucketAcl](#).
- Defina los permisos en un bucket con ACLs, utilizando [PutBucketAcl](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Obtener y establecer una política de lista de control de acceso

Importaciones

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

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

Código de muestra

```
// 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";
}
```

Configuración de buckets de Amazon S3 con el AWS SDK para PHP versión 3

El uso compartido de recursos entre orígenes (CORS) define una manera para que las aplicaciones web de los clientes cargadas en un dominio interactúen con los recursos de un dominio diferente. Gracias a que Amazon S3 admite el CORS, es posible desarrollar aplicaciones web del lado del cliente completas con Amazon S3 y permitir un acceso entre orígenes a los recursos de Amazon S3 de manera selectiva.

Para obtener más información acerca del uso de configuración CORS con un bucket de Amazon S3, consulte [Uso compartido de recursos entre orígenes \(CORS\)](#).

Los siguientes ejemplos muestran cómo:

- Obtenga la configuración CORS para un bucket con [GetBucketCors](#).
- Establecer la configuración CORS para un bucket utilizando [PutBucketCors](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Obtener la configuración CORS

Cree un archivo PHP con el siguiente código. En primer lugar, cree un servicio de cliente AWS.S3, luego llame al método `getBucketCors` y especifique el bucket cuya configuración CORS desea.

El único parámetro necesario es el nombre del bucket seleccionado. Si el bucket tiene actualmente una configuración CORS, Amazon S3 devuelve dicha configuración como un [objeto `CORSRules`](#).

Importaciones

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

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

Código de muestra

```
$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());
}
```

Establecer la configuración CORS

Cree un archivo PHP con el siguiente código. En primer lugar, cree un servicio de cliente de AWS.S3. A continuación, llame al método `putBucketCors` y especifique el bucket cuya configuración CORS se va a establecer y `CORSConfiguration` como [objeto `JSONCORSRules`](#).

Importaciones

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

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

Código de muestra

```
$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());
}
```

Describe cómo utilizar cargas multiparte de Amazon S3 con el AWS SDK para PHP versión 3

Puede cargar objetos de hasta 5 GB en una única operación `PutObject`. Sin embargo, si utiliza los métodos de carga multiparte (por ejemplo, `CreateMultipartUpload`, `UploadPart`, `CompleteMultipartUpload`, `AbortMultipartUpload`), podrá cargar objetos de entre 5 MB y 5 TB.

El siguiente ejemplo muestra cómo:

- Cargue un objeto en Amazon S3 con [ObjectUploader](#).
- Describe cómo crear una carga multiparte para un objeto de Amazon S3 con [MultipartUploader](#).
- Copie objetos de una ubicación de Amazon S3 a otra con [ObjectCopier](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Cargador de objetos

Si no está seguro de si `PutObject` o `MultipartUploader` es la mejor opción para la tarea, utilice `ObjectUploader`. `ObjectUploader` carga un archivo de gran tamaño en Amazon S3 mediante `PutObject` o `MultipartUploader`, en función de lo que sea mejor para el tamaño de la carga.

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

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


Código de muestra

```
$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

Las cargas multiparte están diseñadas para mejorar la experiencia de carga de los objetos más grandes. Estas permiten cargar objetos en partes independientes, en cualquier orden y en paralelo.

Se insta a los clientes de Amazon S3 a utilizar las cargas multiparte para objetos superiores a 100 MB.

ObjetoMultipartUploader

El SDK tiene un objeto `MultipartUploader` especial que simplifica el proceso de carga multiparte.

Importaciones

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

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

Código de muestra

```
$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";
}
```

El cargador crea un generador de datos de la parte, en función del origen que se haya proporcionado y de la configuración, e intenta cargar todas las partes. Si falla la carga de algunas partes, el cargador continúa cargando otras partes hasta que se haya leído todo el origen de datos. A continuación, el cargador vuelve a intentar cargar las partes que han dado error o genera una excepción que contiene información acerca de las partes que no se han podido cargar.

Personalización de una carga multiparte

Puede configurar las opciones personalizadas en las operaciones `CreateMultipartUpload`, `UploadPart` y `CompleteMultipartUpload` ejecutadas por el cargador multiparte mediante devoluciones de llamada transferidas a su constructor.

Importaciones

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

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

Código de muestra

```
// 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';  
    },  
]);
```

Recopilación manual de elementos no utilizados entre cargas de partes

Si se alcanza el límite de memoria al realizar cargas de gran tamaño, puede deberse a las referencias cíclicas generadas por el SDK que el [recolector de elementos no utilizados de PHP](#) todavía no había recopilado cuando se alcanzó el límite de memoria. Si se invoca manualmente el algoritmo de recopilación entre las operaciones, es posible que los ciclos se recopilen antes de alcanzar dicho límite. En el siguiente ejemplo, se invoca el algoritmo de recopilación mediante una devolución de llamada antes de la carga de cada parte. Tenga en cuenta que invocar el recolector de elementos no utilizados conlleva un costo de rendimiento y su uso óptimo dependerá de su caso de uso y su entorno.

```
$uploader = new MultipartUploader($client, $source, [  
    'bucket' => 'your-bucket',  
    'key' => 'your-key',  
    'before_upload' => function(\Aws\Command $command) {  
        gc_collect_cycles();  
    }  
]);
```

Recuperación tras sufrir errores

Cuando se produce un error durante el proceso de carga multiparte, se lanza una excepción `MultipartUploadException`. Esta excepción proporciona acceso al objeto `UploadState`, que contiene información sobre el progreso de la carga multiparte. El objeto `UploadState` puede utilizarse para reanudar una carga que no se ha completado.

Importaciones

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

Código de muestra

```
// 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);
}
```

Si se reanuda una carga a partir de un objeto `UploadState`, se intenta cargar las partes que todavía no se han cargado. El objeto de estado realiza un seguimiento de las partes que faltan, aunque no sean consecutivas. El cargador lee o busca en el archivo de origen facilitado los rangos de bytes que pertenecen a las partes que todavía deben cargarse.

Los objetos `UploadState` se pueden serializar, por lo que también puede reanudar una carga en un proceso diferente. También puede obtener el objeto `UploadState`, incluso si no gestiona una excepción, llamando a `$uploader->getState()`.

Important

Los flujos que se transfieren como un origen a un `MultipartUploader` no se rebobinan automáticamente antes de cargarlos. Si utiliza un flujo en lugar de una ruta de archivo en un bucle similar al del ejemplo anterior, restablezca la variable `$source` dentro del bloque `catch`.

Importaciones

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

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

Código de muestra

```
// 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));
```

Anulación de una carga multiparte

Una carga multiparte se puede anular al recuperar el `UploadId` que se encuentra en el objeto `UploadState` y transferirlo a `abortMultipartUpload`.

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

Cargas multiparte asíncronas

Llamar a `upload()` en el `MultipartUploader` es una solicitud de bloqueo. Si está trabajando en un contexto asíncrono, puede obtener una [promesa](#) (p. 66) para la carga multiparte.

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

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

Código de muestra

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

$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

$promise = $uploader->promise();
```

Configuration

El constructor del objeto `MultipartUploader` acepta los siguientes argumentos:

\$client

Es el objeto `Aws\ClientInterface` que hay que utilizar para ejecutar las transferencias. Debería ser una instancia de `Aws\S3\S3Client`.

\$source

Son los datos de origen que se están cargando. Esto puede ser una ruta o una URL (por ejemplo, `/path/to/file.jpg`), un controlador de recursos (por ejemplo, `fopen('/path/to/file.jpg', 'r')`) o una instancia de un [flujo PSR-7](#).

\$config

Es una matriz asociativa de opciones de configuración para la carga multiparte.

Las siguientes opciones de configuración son válidas:

acl

(string) Es la lista de control de acceso (ACL) para establecer el objeto que se carga. De forma predeterminada, los objetos son privados.

before_complete

(callable) Es la devolución de llamada a invocar antes de la operación `CompleteMultipartUpload`. La devolución de llamada debería tener una firma de la función del tipo `function (Aws\Command $command) {...}`.

before_initiate

(callable) Es la devolución de llamada a invocar antes de la operación `CreateMultipartUpload`. La devolución de llamada debería tener una firma de la función del tipo `function (Aws\Command $command) {...}`.

before_upload

(callable) Es la devolución de llamada a invocar antes de cualquier operación `UploadPart`. La devolución de llamada debería tener una firma de la función del tipo `function (Aws\Command $command) {...}`.

bucket

(string, obligatorio) Es el nombre del bucket al que se está cargando el objeto.

concurrency

(int, predeterminado: `int(5)`) Es el número máximo de operaciones `UploadPart` simultáneas permitido durante la carga multiparte.

clave

(string, obligatorio) Es la clave que se utiliza para el objeto que se está cargando.

part_size

(int, predeterminado: `int(5242880)`) Es el tamaño de la parte, en bytes, que se debe utilizar al realizar una carga multiparte. Debe ser de entre 5 MB y 5 GB, ambos incluidos.

state

(`Aws\Multipart\UploadState`) Es un objeto que representa el estado de la carga multiparte y que se utiliza para reanudar una carga previa. Si se proporciona esta opción, se omiten las opciones `bucket`, `key` y `part_size`.

Copias multiparte

AWS SDK for PHP también incluye un objeto `MultipartCopy` que se utiliza de forma similar al `MultipartUploader`, pero está diseñado para copiar objetos de entre 5 GB y 5 TB en Amazon S3.

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

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

Código de muestra

```
// 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";
}
```

Amazon S3 prefirma de POSTs con el AWS SDK for PHP versión 3

De forma parecida a las URLs prefirmadas, las POSTs prefirmadas le permiten otorgar acceso de escritura a un usuario sin tener que concederle credenciales de AWS. Los formularios POST prefirmados se pueden crear con la ayuda de una instancia de [AwsS3PostObjectV4](#).

Los siguientes ejemplos muestran cómo:

- Obtenga datos para un formulario de carga POST de objetos de S3 utilizando [PostObjectV4](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de PostObjectV4

Para crear una instancia de `PostObjectV4`, debe proporcionar lo siguiente:

- Instancia de `Aws\S3\S3Client`
- bucket
- matriz asociativa de campos de entrada de formularios
- matriz de condiciones de políticas (consulte [Construcción de la política](#) en la Amazon S3 Developer Guide)

- cadena de fecha de vencimiento para la política (opcional, una hora de forma predeterminada).

Importaciones

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

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

Código de muestra

```
$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();
```

URL prefirmada de Amazon S3 con el AWS SDK for PHP versión 3

Puede autenticar determinados tipos de solicitudes pasando la información requerida como parámetros de una cadena de consulta, en lugar de usar el encabezado de autorización HTTP. Esto resulta útil para habilitar el acceso directo por navegador de terceros a sus datos de Amazon S3 privados sin hacer pasar la solicitud por un proxy. La idea es construir una solicitud "prefirmada" y codificarla como una URL que pueda recuperar el navegador de un usuario final. Además, puede limitar una solicitud prefirmada especificando una fecha de vencimiento.

Los siguientes ejemplos muestran cómo:

- Describe cómo crear una URL prefirmada para obtener un objeto de S3 mediante [createPresignedRequest](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Crear una solicitud prefirmada

Para obtener la URL prefirmada de un objeto de Amazon S3 puede usar el método `Aws\S3\S3Client::createPresignedRequest()`. Este método acepta un objeto `Aws\CommandInterface` y una marca de tiempo caducada y devuelve un objeto `Psr\Http\Message\RequestInterface` prefirmado. Para obtener la URL prefirmada del objeto, utilice el método `getUri()` de la solicitud.

La situación más habitual consiste en crear una URL prefirmada para obtener un objeto.

Importaciones

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

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

Código de muestra

```
$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');
```

Crear una URL prefirmada

Puede crear URLs prefirmadas para cualquier operación de Amazon S3 con el método `getCommand` para crear un objeto comando y, a continuación, llamar al método `createPresignedRequest()` con el comando. Cuando al final se envía la solicitud, asegúrese de utilizar el mismo método y los mismos encabezados que en la solicitud devuelta.

Código de muestra

```
//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();
```

Obtener la URL a un objeto

Si solo necesita la URL pública para un objeto almacenado en un bucket de Amazon S3, puede usar el método `Aws\S3\S3Client::getObjectUrl()`. Este método devuelve una URL sin firmar al bucket y clave indicados.

Código de muestra

```
//Getting the URL to an object  
$url = $s3Client->getObjectUrl('my-bucket', 'my-key');
```

Important

La URL devuelta por este método no está validada para garantizar que el bucket o la clave existan y este método tampoco asegura que el objeto permita el acceso sin autenticar.

Uso de un bucket de Amazon S3 como un host web estático con el AWS SDK for PHP versión 3

Puede alojar un sitio web estático en Amazon S3. Para obtener más información, consulte [Alojamiento de un sitio web estático en S3](#).

El siguiente ejemplo muestra cómo:

- Obtenga la configuración de sitio web para un bucket con [GetBucketWebsite](#).
- Establezca la configuración de sitio web para un bucket con [PutBucketWebsite](#).
- Elimine la configuración de sitio web de un bucket con [DeleteBucketWebsite](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS. Consulte [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#).

Obtener, establecer y eliminar la configuración de sitio web para un bucket

Importaciones

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

Código de muestra

```
$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";
}
```

Uso de políticas de bucket de Amazon S3 con el AWS SDK para PHP versión 3

Puede utilizar una política de bucket para conceder permisos a los recursos de Amazon S3. Para obtener más información, consulte [Uso de políticas de bucket y usuario](#).

El siguiente ejemplo muestra cómo:

- Devolver la política para un bucket especificado utilizando [GetBucketPolicy](#).
- Reemplazar una política en un bucket con [PutBucketPolicy](#).
- Elimine una política de un bucket con [DeleteBucketPolicy](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Obtener, eliminar y sustituir una política en un bucket

Importaciones

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

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

Código de muestra

```
$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 (string) $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";
}
```

Con el punto de acceso de S3 ARNs el AWS SDK para PHP versión 3

S3 introdujo los puntos de acceso, una nueva forma de interactuar con los buckets de S3. Los puntos de acceso pueden tener políticas y configuraciones únicas aplicadas a ellos en lugar de directamente al bucket. El AWS SDK para PHP le permite utilizar el punto de acceso ARNs en el campo del bucket para operaciones de API de en lugar de especificar explícitamente el nombre del bucket. Más información sobre cómo encontrar los puntos de acceso de S3 y el trabajo de ARNs

[here <https://docs.aws.amazon.com/AmazonS3/latest/dev/using-access-points.html> _</problematic>](https://docs.aws.amazon.com/AmazonS3/latest/dev/using-access-points.html)

:. El siguiente ejemplo muestra cómo:

- Use [GetObject](#) con un ARN de punto de acceso para obtener un objeto de un bucket.
- Use [PutObject](#) con un ARN de punto de acceso para agregar un objeto a un bucket.
- Configure el cliente de S3 para que utilice la región ARN en lugar de la región de cliente.

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Importaciones

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

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

Get Object

En primer lugar, cree un servicio cliente AWS.S3 que especifique la región de AWS y la versión. Luego llame al método `getObject` con su clave y un ARN de punto de acceso de S3 en el campo `Bucket`, que obtendrá el objeto del bucket asociado con ese punto de acceso.

Código de muestra

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);
$result = $s3->getObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key'    => 'MyKey'
]);
```

Colocar un objeto en un bucket

En primer lugar, cree un servicio cliente AWS.S3 que especifique la región de AWS y la versión. A continuación, llame al método `putObject` con la clave, el cuerpo o el archivo de origen deseados y un ARN de punto de acceso de S3 en el campo `Bucket`, que colocará el objeto en el bucket asociado con ese punto de acceso.

Código de muestra

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);
$result = $s3->putObject([
    'Bucket' => 'arn:aws:s3:us-west-2:123456789012:accesspoint:endpoint-name',
    'Key'    => 'MyKey',
    'Body'   => 'MyBody'
]);
```

Configure el cliente de S3 para que utilice la región ARN en lugar de la región de cliente

Cuando se utiliza un ARN de punto de acceso de S3 en una operación de cliente de S3, de forma predeterminada el cliente se asegurará de que la región ARN coincida con la región de cliente, produciendo una excepción si no lo hace. Este comportamiento se puede cambiar para aceptar la región ARN sobre la región cliente estableciendo la opción de configuración `use_arn_region` en `true`. De forma predeterminada, la opción se establece en `false`.

Código de muestra

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

El cliente también comprobará una variable de entorno y una opción de archivo de configuración, en el siguiente orden de prioridad:

1. La opción de cliente `use_arn_region`, como en el ejemplo anterior.
2. La variable de entorno `AWS_S3_USE_ARN_REGION`

```
export AWS_S3_USE_ARN_REGION=true
```

1. La variable de configuración `s3_use_arn_region` en el archivo de configuración compartida de AWS (de forma predeterminada en `~/.aws/config`).

```
[default]
s3_use_arn_region = true
```

Administración de secretos mediante la API de Secrets Manager y el AWS SDK for PHP versión 3

AWS Secrets Manager almacena y administra secretos compartidos como, por ejemplo, contraseñas, claves de API y credenciales de bases de datos. Con el servicio Secrets Manager, los desarrolladores pueden sustituir credenciales codificadas de forma rígida en el código implementado por una llamada integrada a Secrets Manager.

Secrets Manager es compatible de forma nativa con la rotación de credenciales programada automática para las bases de datos de Amazon Relational Database Service (Amazon RDS), lo que aumenta la seguridad de las aplicaciones. Secrets Manager también puede rotar fácilmente secretos para otras bases de datos y servicios de terceros utilizando AWS Lambda para implementar detalles específicos de los servicios.

Los siguientes ejemplos muestran cómo:

- Crear un secreto con [CreateSecret](#).
- Recuperar un secreto con [GetSecretValue](#).
- Obtener una lista de todos los secretos almacenados por Secrets Manager con [ListSecrets](#).
- Obtenga información detallada sobre un secreto especificado mediante [DescribeSecret](#).
- Actualizar un secreto especificado con [PutSecretValue](#).
- Configure una rotación de secretos mediante [RotateSecret](#).
- Marcar un secreto para eliminarlo con [DeleteSecret](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de un secreto en Secrets Manager

Para crear un secreto en Secrets Manager, utilice la operación [CreateSecret](#).

En este ejemplo, un nombre de usuario y una contraseña se almacenan en una cadena JSON.

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Recuperación de un secreto de Secrets Manager

Para recuperar el valor de un secreto almacenado en Secrets Manager, utilice la operación [GetSecretValue](#).

En este ejemplo, `secret` es una cadena que contiene el valor almacenado. Si se le llama en el secreto que creamos anteriormente, este ejemplo muestra `[{"username\":"<USERNAME>\","password\":"<PASSWORD>"}]`. Utilice `json.loads` para obtener acceso a los valores indexados.

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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 == 'InternalServerErrorException') {
        // 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;
```

Listado de los secretos almacenados en Secrets Manager

Obtenga una lista de todos los secretos que Secrets Manager almacena mediante la operación [ListSecrets](#).

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Recuperación de los detalles de un secreto

Los secretos almacenados contienen metadatos sobre las reglas de rotación, la fecha en que se modificaron o se tuvo acceso a ellos por última vez, las etiquetas creadas por el usuario y el nombre de recurso de Amazon (ARN). Para obtener los detalles de un secreto especificado almacenado en Secrets Manager, utilice la operación [DescribeSecret](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\SecretsManager\SecretsManagerClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Actualización del valor del secreto

Para almacenar un nuevo valor de secreto cifrado en Secrets Manager, utilice la operación [PutSecretValue](#).

De esta forma, se crea una versión nueva del secreto. Si ya existe una versión del secreto, añada el parámetro `VersionStages` con el valor existente en `AWSCURRENT` para asegurarse de que se utiliza el valor nuevo al recuperar el valor.

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Rotación del valor de un secreto existente en Secrets Manager

Para rotar el valor de un secreto existente almacenado en Secrets Manager, utilice una función de rotación Lambda y la operación [RotateSecret](#).

Antes de comenzar, cree una función de Lambda para rotar el secreto. El [catálogo de ejemplos de código de AWS](#) contiene varios ejemplos de código de Lambda para la rotación de credenciales de base de datos de Amazon RDS.

Note

Para obtener más información acerca de la rotación de secretos, consulte [Rotación de sus secretos de AWS Secrets Manager](#) en la AWS Secrets Manager User Guide.

Después de configurar la función de Lambda, configure una rotación de secretos nueva.

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Al configurar una rotación, puede implementar una rotación utilizando la operación [RotateSecret](#).

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Eliminación de un secreto de Secrets Manager

Para eliminar un secreto especificado de Secrets Manager, utilice la operación [DeleteSecret](#). Para evitar la eliminación accidental de un secreto, se añade automáticamente una marca `DeletionDate` al secreto que especifica una ventana de tiempo de recuperación en la que puede invertir la eliminación. Si no se especifica la fecha para el intervalo de recuperación, el periodo de tiempo predeterminado es de 30 días.

Importaciones

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Información relacionada

Los ejemplos del AWS SDK for PHP utilizan las siguientes operaciones REST de la AWS Secrets Manager API Reference:

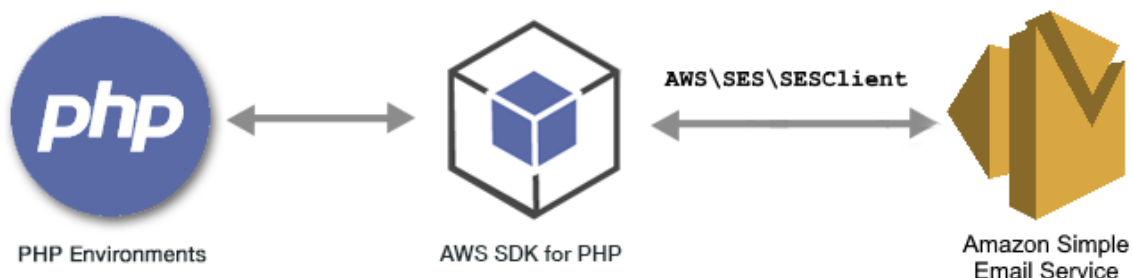
- [CreateSecret](#)
- [GetSecretValue](#)
- [ListSecrets](#)
- [DescribeSecret](#)
- [PutSecretValue](#)
- [RotateSecret](#)
- [DeleteSecret](#)

Para obtener más información acerca de cómo utilizar AWS Secrets Manager, consulte la [Guía del usuario de AWS Secrets Manager](#).

Ejemplos de Amazon SES que utilizan el AWS SDK for PHP versión 3

Amazon Simple Email Service (Amazon SES) es una plataforma de correo electrónico que ofrece un método sencillo y rentable de enviar y recibir correos electrónicos utilizando sus propios dominios y

direcciones de correo electrónico. Para obtener más información sobre Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- Verificación de identidades de correo electrónico mediante la API de Amazon SES y el AWS SDK for PHP versión 3 (p. 255)
- Creación de plantillas de correo electrónico personalizadas mediante la API de Amazon SES y AWS SDK for PHP versión 3 (p. 260)
- Administración de filtros de correo electrónico utilizando la API de Amazon SES y el AWS SDK for PHP versión 3 (p. 264)
- Creación y administración de reglas de correo electrónico mediante la API de Amazon SES y el AWS SDK for PHP versión 3 (p. 267)
- Monitorización de la actividad de envío utilizando la API de Amazon SES y el AWS SDK for PHP versión 3 (p. 273)
- Autorización de remitentes mediante la API de Amazon SES y el AWS SDK for PHP versión 3 (p. 275)

Verificación de identidades de correo electrónico mediante la API de Amazon SES y el AWS SDK for PHP versión 3

La primera vez que se utiliza la cuenta de Amazon Simple Email Service (Amazon SES), todos los remitentes y destinatarios deben verificarse en la misma región de AWS a la que se envían los mensajes de correo electrónico. Para obtener más información sobre el envío de mensajes de correo electrónico, consulte [Envío de correo electrónico con Amazon SES](#).

Los siguientes ejemplos muestran cómo:

- Verifique una dirección de correo electrónico utilizando [VerifyEmailIdentity](#).
- Describe cómo verificar un dominio de correo electrónico mediante [VerifyDomainIdentity](#).
- Muestre todas las direcciones de correo electrónico con [ListIdentities](#).
- Muestre todos los dominios de correo electrónico mediante [ListIdentities](#).
- Elimine una dirección de correo electrónico utilizando [DeleteIdentity](#).
- Describe cómo eliminar un dominio de correo electrónico mediante [DeleteIdentity](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Verificación de direcciones de correo electrónico

Amazon SES solo puede enviar mensajes de correo electrónico desde direcciones de correo electrónico o dominios verificados. Al verificar una dirección de correo electrónico, demuestra que usted es el propietario de esa dirección y que desea permitir que Amazon SES envíe mensajes de correo electrónico desde esa dirección.

Al ejecutar el siguiente ejemplo de código, Amazon SES envía un mensaje de correo electrónico a la dirección especificada. Cuando usted (o el destinatario del mensaje de correo electrónico) hagan clic en el enlace del mensaje, la dirección se habrá verificado.

Para añadir una dirección de correo electrónico a su cuenta de Amazon SES, utilice la operación [VerifyEmailIdentity](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Verificar una dirección de correo electrónico

Amazon SES solo puede enviar mensajes de correo electrónico desde direcciones de correo electrónico o dominios verificados. Al verificar un dominio, demuestra que es el propietario de ese dominio. Si verifica un

dominio, permite a Amazon SES enviar correo electrónico desde cualquier dirección de correo electrónico de dicho dominio.

Cuando ejecute el siguiente ejemplo de código, Amazon SES le proporcionará un token de verificación. Debe añadir el token a la configuración de DNS del dominio. Para obtener más información, consulte la sección [Verificación de un dominio con Amazon SES](#) en la Amazon SES Developer Guide.

Para añadir un dominio de envío a su cuenta de Amazon SES, utilice la operación [VerifyDomainIdentity](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Lista de direcciones de correo electrónico

Para recuperar una lista de direcciones de correo electrónico enviada en la región de AWS actual, independientemente del estado de verificación, utilice la operación [ListIdentities](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Lista de dominios de correo electrónico

Para recuperar una lista de dominios de correo electrónico enviada en la región de AWS actual, independientemente del estado de verificación, utilice la operación [ListIdentities](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Eliminación de una dirección de correo electrónico

Para eliminar una dirección de correo electrónico verificada de la lista de identidades, utilice la operación [DeleteIdentity](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;
```

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

Código de muestra

```
$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";
}
```

Eliminación de un dominio de correo electrónico

Para eliminar un dominio de correo electrónico verificado de la lista de identidades verificadas, utilice la operación [DeleteIdentity](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Creación de plantillas de correo electrónico personalizadas mediante la API de Amazon SES y AWS SDK for PHP versión 3

Amazon Simple Email Service (Amazon SES) permite enviar mensajes de correo electrónico personalizados para cada destinatario mediante el uso de plantillas. Las plantillas incluyen una línea de asunto y las partes de texto y HTML del cuerpo del correo electrónico. Las secciones de asunto y cuerpo también pueden contener valores únicos personalizados para cada destinatario.

Para obtener más información, consulte la sección de [envío de mensajes de correo electrónico personalizados con Amazon SES](#) en la Amazon SES Developer Guide.

Los siguientes ejemplos muestran cómo:

- Cree una plantilla de correo electrónico mediante [CreateTemplate](#).
- Muestre todas las plantillas de correo electrónico mediante [ListTemplates](#).
- Recupere una plantilla de correo electrónico mediante [GetTemplate](#).
- Actualice una plantilla de correo electrónico mediante [UpdateTemplate](#).
- Describe cómo eliminar una plantilla de correo electrónico mediante [DeleteTemplate](#).
- Enviar un correo electrónico con una plantilla utilizando [SendTemplatedEmail](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Creación de una plantilla de correo electrónico

Para crear una plantilla para enviar mensajes de correo electrónico personalizados, utilice la operación [CreateTemplate](#). La plantilla la puede utilizar cualquier cuenta autorizada a enviar mensajes en la región de AWS a la que la plantilla se añada.

Note

Amazon SES no valida su HTML, por lo que debe asegurarse de que `HtmlPart` es válido antes de enviar un correo electrónico.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Obtener una plantilla de correo electrónico

Para ver el contenido de una plantilla de correo electrónico existente, incluida la línea de asunto, el cuerpo en HTML y el texto sin formato, utilice la operación [GetTemplate](#). Solo se requiere TemplateName.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Enumerar todas las plantillas de correo electrónico

Para recuperar una lista de todas las plantillas de correo electrónico asociadas a su cuenta de AWS en la región de AWS actual, utilice la operación [ListTemplates](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Actualizar una plantilla de correo electrónico

Para cambiar el contenido de una plantilla de correo electrónico específica incluida la línea de asunto, el cuerpo en HTML y el texto sin formato, utilice la operación [UpdateTemplate](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Eliminar una plantilla de correo electrónico

Para eliminar una plantilla de correo electrónico específica, utilice la operación [DeleteTemplate](#). Lo único que necesita es el `TemplateName`.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Enviar un mensaje de correo electrónico con una plantilla

Para utilizar una plantilla para enviar un correo electrónico a los destinatarios, utilice la operación [SendTemplatedEmail](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Administración de filtros de correo electrónico utilizando la API de Amazon SES y el AWS SDK for PHP versión 3

Además de enviar correos electrónicos, también puede recibir correo electrónico con Amazon Simple Email Service (Amazon SES). Un filtro de direcciones IP le permite especificar opcionalmente si desea aceptar o rechazar el correo que procede de una dirección IP o de un intervalo de direcciones IP. Para obtener más información, consulte [Administración de filtros de direcciones IP para recepción de correo electrónico de Amazon SES](#).

Los siguientes ejemplos muestran cómo:

- Describe cómo crear un filtro de correo electrónico con [CreateReceiptFilter](#).
- Obtenga una lista de todos los filtros de correo electrónico utilizando [ListReceiptFilters](#).
- Elimine un filtro de correo electrónico utilizando [DeleteReceiptFilter](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Creación de un filtro de correo electrónico

Para permitir o bloquear correos electrónicos desde una dirección IP específica, utilice la operación [CreateReceiptFilter](#). Proporcione la dirección o el rango de direcciones IP y un nombre único para identificar este filtro.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```


Enumerar todos los filtros de correo electrónico

Para obtener una lista de los filtros de direcciones IP asociados a su cuenta de AWS en la región de AWS actual, utilice la operación [ListReceiptFilters](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Eliminación de un filtro de correo electrónico

Para eliminar un filtro existente para una dirección IP específica, utilice la operación [DeleteReceiptFilter](#). Proporcione el nombre del filtro único para identificar el filtro de recepción que desea eliminar.

Si necesita cambiar el rango de direcciones que se filtran, puede eliminar una filtro de recepción y crear uno nuevo.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Creación y administración de reglas de correo electrónico mediante la API de Amazon SES y el AWS SDK for PHP versión 3

Además de enviar correos electrónicos, también puede recibir correo electrónico con Amazon Simple Email Service (Amazon SES). Las reglas de recepción le permiten especificar qué hace Amazon SES con el correo electrónico que reciba para las direcciones de correo electrónico o dominios de su propiedad. Una regla puede enviar correo electrónico a otros servicios de AWS, incluidos, con carácter meramente enunciativo, Amazon S3, Amazon SNS o AWS Lambda.

Para obtener más información, consulte la sección [Administración de conjuntos de reglas de recepción para recepción de correo electrónico de Amazon SES](#) y [Administración de reglas de recepción para recepción de correo electrónico de Amazon SES](#).

Los siguientes ejemplos muestran cómo:

- Creación de un conjunto de reglas de recepción utilizando [CreateReceiptRuleSet](#).
- Creación de una regla de recepción utilizando [CreateReceiptRule](#).
- Describa un conjunto de reglas de recepción utilizando [DescribeReceiptRuleSet](#).
- Describa una regla de recepción mediante [DescribeReceiptRule](#).
- Obtenga una lista de todos los conjuntos de reglas de recepción utilizando [ListReceiptRuleSets](#).
- Actualice una regla de recepción mediante [UpdateReceiptRule](#).
- Elimine una regla de recepción mediante [DeleteReceiptRule](#).
- Elimine un conjunto de reglas de recepción mediante [DeleteReceiptRuleSet](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Creación de un conjunto de reglas de recepción

Un conjunto de reglas de recepción contiene una colección de reglas de recepción. Debe tener al menos un conjunto de reglas de recepción asociadas a su cuenta para poder crear una regla de recepción.

Para crear un conjunto de reglas de recepción, proporcione un RuleSetName único y utilice la operación [CreateReceiptRuleSet](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Creación de una regla de recepción

Controle el correo electrónico entrante añadiendo una regla de recepción a un conjunto de reglas de recepción existente. En este ejemplo se muestra cómo crear una regla de recepción que envía los mensajes entrantes a un bucket de Amazon S3, pero también puede enviar mensajes a Amazon SNS y AWS Lambda. Para crear una regla de recepción, proporcione una regla y RuleSetName a la operación [CreateReceiptRule](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Descripción de un conjunto de reglas de recepción

Una vez por cada segundo, devuelva los detalles del conjunto de reglas de recepción especificado. Para utilizar la operación [DescribeReceiptRuleSet](#), proporcione el RuleSetName.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Descripción de una regla de recepción

Devuelva los detalles de una regla de recepción especificada. Para utilizar la operación [DescribeReceiptRule](#), proporcione los valores de RuleName y RuleSetName.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Enumeración de todos los conjuntos de reglas de recepción

Para enumerar los conjuntos de reglas de recepción que existen en su cuenta de AWS en la región de AWS actual, utilice la operación [ListReceiptRuleSets](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Actualización de una regla de recepción

En este ejemplo se muestra cómo actualizar una regla de recepción que envía los mensajes entrantes a una función de AWS Lambda, pero también puede enviar mensajes a Amazon SNS y Amazon S3. Para utilizar la operación [UpdateReceiptRule](#), proporcione la nueva regla de recepción y el RuleSetName.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Eliminación de un conjunto de reglas de recepción

Elimine un determinado conjunto de reglas de recepción que no esté deshabilitado actualmente. Esta acción también elimina todas las reglas de recepción que contiene. Para eliminar un conjunto de reglas de recepción, proporcione RuleSetName a la operación [DeleteReceiptRuleSet](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Eliminación de una regla de recepción

Para eliminar una regla de recepción especificada, proporcione RuleName y RuleSetName a la operación [DeleteReceiptRule](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Monitorización de la actividad de envío utilizando la API de Amazon SES y el AWS SDK for PHP versión 3

Amazon Simple Email Service (Amazon SES) ofrece métodos para monitorizar la actividad de envío. Le recomendamos que implemente estos métodos para que pueda realizar un seguimiento de medidas importantes, como, por ejemplo, las tasas de rebotes, reclamaciones y rechazos de su cuenta. Unas tasas de rebotes y reclamaciones demasiado altas pueden comprometer su capacidad de enviar correos electrónicos utilizando Amazon SES.

Los siguientes ejemplos muestran cómo:

- Compruebe su cuota de envío con [GetSendQuota](#).
- Monitorice su actividad de envío con [GetSendStatistics](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Comprobar su cuota de envío

Tiene un límite de enviar solo una cantidad determinada de mensajes en un único periodo de 24 horas. Para comprobar la cantidad de mensajes que aún puede enviar, utilice la operación [GetSendQuota](#). Para obtener más información, consulte [Administrar sus límites de envío de Amazon SES](#).

Importaciones

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



```
use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Monitorización de la actividad de envío

Para recuperar métricas de mensajes que haya enviado en las últimas dos semanas, utilice la operación [GetSendStatistics](#). En este ejemplo, se devuelve el número de intentos de entrega, rebotes, reclamaciones y mensajes rechazados en incrementos de 15 minutos.

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Autorización de remitentes mediante la API de Amazon SES y el AWS SDK for PHP versión 3

Para permitir que otra cuenta de AWS, usuario de AWS Identity and Access Management o servicio de AWS envíe mensajes de correo electrónico a través de Amazon Simple Email Service (Amazon SES) en su nombre, debe crear una política de autorización de envío. Se trata de un documento JSON que se asocia a una identidad de su propiedad.

La política enumera de forma expresa a quién permite enviar para dicha identidad y en qué condiciones. No se permite enviar mensajes de correo electrónico a todos los remitentes excepto usted y las entidades a las que conceda permiso explícitamente en la política. Una identidad puede no tener ninguna política, una política o varias políticas asociadas. También puede tener una política con varias instrucciones para conseguir el efecto de varias políticas.

Para obtener más información, consulte [Using Sending Authorization with Amazon SES](#).

Los siguientes ejemplos muestran cómo:

- Crear un remitente autorizado utilizando [PutIdentityPolicy](#).
- Recupere políticas para un remitente autorizado mediante [GetIdentityPolicies](#).
- Obtenga una lista de remitentes autorizados mediante [ListIdentityPolicies](#).
- Revoque el permiso para un remitente autorizado mediante [DeleteIdentityPolicy](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Para obtener más información acerca del uso de Amazon SES, consulte la [Guía para desarrolladores de Amazon SES](#).

Creación de un remitente autorizado

Para autorizar a otra cuenta de AWS para enviar mensajes de correo electrónico en su nombre, utilice una política de identidad para añadir o actualizar la autorización para enviar mensajes de correo electrónico desde sus direcciones de correo electrónico verificadas o dominios. Para crear una política de identidad, utilice la operación [PutIdentityPolicy](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";
}
```

Recuperación de políticas para un remitente autorizado

Devuelva las políticas de autorización de envío que estén asociadas a una identidad de correo electrónico o de dominio específicas. Para obtener la autorización de envío de una determinada dirección de correo electrónico o dominio, utilice la operación [GetIdentityPolicy](#).

Importaciones

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

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Lista de remitentes autorizados

Para enumerar las políticas de autorización de envío asociadas a una identidad de correo electrónico o de dominio específicas en la región de AWS actual, utilice la operación [ListIdentityPolicies](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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";  
}
```

Revocación del permiso para un remitente autorizado

Para eliminar la autorización de envío de otra cuenta de AWS para enviar correos electrónicos con una identidad de correo electrónico o de dominio, elimine la política de identidad asociada con la operación [DeleteIdentityPolicy](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Ses\SesClient;  
use Aws\Exception\AwsException;
```

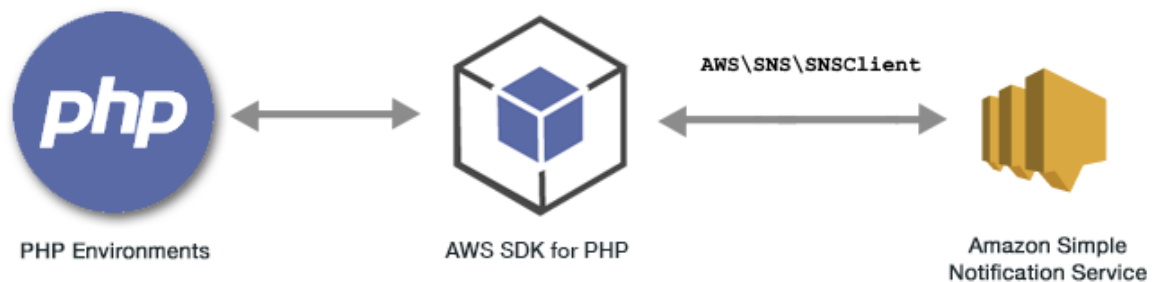
Código de muestra

```
$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";  
}
```

Ejemplos de Amazon SNS que utilizan el AWS SDK for PHP versión 3

Amazon Simple Notification Service (Amazon SNS) es un servicio web que coordina y administra la entrega o el envío de mensajes a los puntos de enlace o clientes suscritos.

En Amazon SNS existen dos tipos de clientes: editores (también conocidos como productores) y suscriptores (también conocidos como consumidores).



Los publicadores se comunican de forma asíncrona con los suscriptores generando y enviando un mensaje a un tema, que es un punto de acceso lógico y un canal de comunicación. Los suscriptores

(servidores web, direcciones de correo electrónico, colas de Amazon SQS o funciones de AWS Lambda) consumen o reciben el mensaje o la notificación por medio de uno de los protocolos admitidos (Amazon SQS, HTTP/HTTPS URLs, correo electrónico, AWS SMS o Lambda) cuando están suscritos al tema.

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- [Administración de temas en Amazon SNS con el AWS SDK for PHP versión 3 \(p. 279\)](#)
- [Administración de suscripciones en Amazon SNS con el AWS SDK for PHP versión 3 \(p. 282\)](#)
- [Envío de mensajes SMS en Amazon SNS con el AWS SDK for PHP versión 3 \(p. 288\)](#)

Administración de temas en Amazon SNS con el AWS SDK for PHP versión 3

Para enviar notificaciones a Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS URLs, correo electrónico, AWS SMS o AWS Lambda, primero debe crear un tema que administre la entrega de mensajes a los suscriptores de ese tema.

En lo relativo al patrón de diseño de observador, un tema es como el asunto. Una vez que se crea un tema, se añaden los suscriptores que reciben notificaciones automáticas cuando se publica un mensaje en el tema.

Obtenga más información sobre la suscripción a temas en la sección de [Administración de suscripciones en Amazon SNS con AWS SDK para PHP versión 3 \(p. 282\)](#).

Los siguientes ejemplos muestran cómo:

- Cree un tema para publicar notificaciones en utilizando [CreateTopic](#).
- Devuelve una lista de los temas del solicitante mediante [ListTopics](#).
- Elimine un tema y todas sus suscripciones con [DeleteTopic](#).
- Devolver todas las propiedades de un tema con [GetTopicAttributes](#).
- Permitir que el propietario de un tema establezca un atributo del tema en un nuevo valor utilizando [SetTopicAttributes](#).

Para obtener más información sobre el uso de Amazon SNS, consulte [Atributos de tema de Amazon SNS para el estado de entrega del mensaje](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Creación de un tema

Para crear un tema, utilice la operación [CreateTopic](#).

Cada nombre de tema de su cuenta de AWS debe ser exclusivo.

Importaciones

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

```
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Lista de temas

Para obtener una lista de hasta 100 temas existentes en la región de AWS actual, utilice la operación [ListTopics](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Eliminación de un tema

Para eliminar un tema existente y todas sus suscripciones, utilice la operación [DeleteTopic](#).

Todos los mensajes que no se han entregado a los suscriptores también se eliminarán.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Obtener de atributos de los temas

Para recuperar las propiedades de un único tema existente, utilice la operación [GetTopicAttributes](#).

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Establecer de atributos de los temas

Para actualizar las propiedades de un único tema existente, utilice la operación [SetTopicAttributes](#).

Solo puede establecer los atributos Policy, DisplayName y DeliveryPolicy.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Administración de suscripciones en Amazon SNS con el AWS SDK for PHP versión 3

Utilice los temas de Amazon Simple Notification Service (Amazon SNS) para enviar notificaciones a Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, direcciones de correo electrónico, AWS Server Migration Service (AWS SMS) o AWS Lambda.

Las suscripciones se adjuntan a un tema que administra el envío de mensajes a los suscriptores. Obtenga más información sobre la creación de temas en [Administración de temas en Amazon SNS con el AWS SDK para PHP versión 3](#) (p. 279).

Los siguientes ejemplos muestran cómo:

- Suscribirse a un tema existente mediante [Subscribe](#).
- Verifique una suscripción mediante [ConfirmSubscription](#).
- Obtener una lista de las suscripciones existentes con [ListSubscriptionsByTopic](#).
- Eliminar una suscripción mediante [Unsubscribe](#).
- Enviar un mensaje a todos los suscriptores de un tema, mediante [Publish](#).

Para obtener más información acerca de cómo utilizar Amazon SNS, consulte [Uso de Amazon SNS para la mensajería entre sistemas](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Suscripción de una dirección de correo electrónico a un tema

Para iniciar una suscripción a una dirección de correo electrónico, utilice la operación [Suscribe](#).

Puede utilizar el método de suscripción para suscribir a varios puntos de enlace diferentes a un tema de Amazon SNS, en función de los valores utilizados para los parámetros pasados. Esto se muestra en otros ejemplos en este tema.

En este ejemplo, el punto de enlace es una dirección de correo electrónico. Se envía un token de confirmación a este correo electrónico. Compruebe la suscripción con este token de confirmación en un plazo de tres días desde la recepción.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Suscripción de un punto de enlace de aplicación a un tema

Para iniciar una suscripción a una aplicación web, utilice la operación [Suscribe](#).

Puede utilizar el método de suscripción para suscribir a varios puntos de enlace diferentes a un tema de Amazon SNS, en función de los valores utilizados para los parámetros pasados. Esto se muestra en otros ejemplos en este tema.

En este ejemplo, el punto de enlace es una URL. Se envía un token de confirmación a esta dirección web. Compruebe la suscripción con este token de confirmación en un plazo de tres días desde la recepción.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Suscripción de una función de Lambda a un tema

Para iniciar una suscripción a una función de Lambda, utilice la operación [Suscribe](#).

Puede utilizar el método de suscripción para suscribir a varios puntos de enlace diferentes a un tema de Amazon SNS, en función de los valores utilizados para los parámetros pasados. Esto se muestra en otros ejemplos en este tema.

En este ejemplo, el punto de enlace es una función de Lambda. Se envía un token de confirmación a esta función de Lambda. Compruebe la suscripción con este token de confirmación en un plazo de tres días desde la recepción.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Suscripción de un mensaje SMS a un tema

Para enviar mensajes SMS a varios números de teléfono al mismo tiempo, suscriba cada número a un tema.

Para iniciar una suscripción a un número de teléfono, utilice la operación [Suscribe](#).

Puede utilizar el método de suscripción para suscribir a varios puntos de enlace diferentes a un tema de Amazon SNS, en función de los valores utilizados para los parámetros pasados. Esto se muestra en otros ejemplos en este tema.

En este ejemplo, el punto de enlace es un número de teléfono en formato E.164, un estándar utilizado para las telecomunicaciones internacionales.

Se envía un token de confirmación a este número de teléfono. Compruebe la suscripción con este token de confirmación en un plazo de tres días desde la recepción.

Para una conocer una manera alternativa de enviar mensajes SMS con Amazon SNS, consulte [Envío de mensajes SMS en Amazon SNS con AWS SDK para PHP versión 3 \(p. 288\)](#).

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Confirmación de la suscripción a un tema

Para crear una suscripción real, el propietario del punto de enlace debe reconocer la intención de recibir mensajes del tema utilizando un token enviado cuando se establece inicialmente una suscripción, tal y como se ha descrito anteriormente. Los tokens de confirmación son válidos durante tres días. Después de tres días, puede volver a enviar un token mediante la creación de una nueva suscripción.

Para confirmar una suscripción, utilice la operación [ConfirmSubscription](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Lista de suscripciones a un tema

Para obtener una lista de hasta 100 suscripciones existentes en una región de AWS determinada, utilice la operación [ListSubscriptions](#).

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Cancelación de la suscripción a un tema

Para eliminar un punto de enlace suscrito a un tema, utilice la operación [Unsubscribe](#).

Si la suscripción requiere autenticación para su eliminación, solo el propietario de la suscripción o el propietario del tema pueden cancelar la suscripción y se requiere una firma de AWS. Si la llamada de cancelación de suscripción no requiere autenticación y el solicitante no es el propietario de la suscripción, se envía un último mensaje de cancelación al punto de enlace.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Publicación de un mensaje en un tema Amazon SNS

Para entregar un mensaje a cada punto de enlace que está suscrito a un tema Amazon SNS, utilice la operación [Publish](#).

Cree un objeto que contenga los parámetros para la publicación de un mensaje, incluido el texto del mensaje y el nombre de recurso de Amazon (ARN) del tema Amazon SNS.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Envío de mensajes SMS en Amazon SNS con el AWS SDK for PHP versión 3

Puede utilizar Amazon Simple Notification Service (Amazon SNS) para enviar mensajes de texto o mensajes SMS a dispositivos habilitados para recibir SMS. Dispone de la capacidad de enviar un mensaje directamente a un número de teléfono o de enviar un mensaje a varios números de teléfono a la vez suscribiendo dichos números de teléfono a un tema y enviando el mensaje al tema.

Utilice Amazon SNS para especificar las preferencias de los mensajes SMS, como, por ejemplo, cómo se optimizan las entregas (costos o entregas fiables), el límite de gasto mensual, cómo se registran las entregas de mensajes y si desea suscribirse a informes de uso de SMS diarios. Estas preferencias se recuperan y se establecen como atributos de SMS para Amazon SNS.

Cuando envíe un mensaje SMS, especifique el número de teléfono usando la formato E.164. E.164 es un estándar de estructura de número de teléfono utilizada para las telecomunicaciones internacionales. Los números de teléfono que aplican este formato pueden tener un máximo de 15 dígitos y van prefijados con el carácter (+) y el código de país. Por ejemplo, un número de teléfono de los EE. UU. en formato E.164 se mostraría como +1001XXX5550100.

Los siguientes ejemplos muestran cómo:

- Recupere la configuración predeterminada para enviar mensajes SMS desde su cuenta con [GetSMSAttributes](#).
- Actualice la configuración predeterminada para el envío de mensajes SMS desde su cuenta con [SetSMSAttributes](#).
- Obtenga información sobre si un determinado propietario de número de teléfono ha desactivado la recepción de mensajes SMS desde su cuenta con [CheckIfPhoneNumberIsOptedOut](#).
- Obtenga una lista de los números de teléfono en los que el propietario ha desactivado la recepción de mensajes SMS desde su cuenta con [ListPhoneNumberOptedOut](#).

- Envíe un mensaje de texto (mensaje SMS) directamente a un número de teléfono usando [Publish](#).

Para obtener más información acerca de cómo utilizar Amazon SNS, consulte [Uso de Amazon SNS para las notificaciones de usuario con un número de teléfono móvil como suscriptor \(envío de SMS\)](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Obtención de atributos de SMS

Para recuperar la configuración predeterminada de mensajes SMS, utilice la operación [GetSMSAttributes](#).

Este ejemplo obtiene el atributo `DefaultSMSType`. Este atributo controla si se envían mensajes SMS como `Promotional`, que optimiza la entrega de mensajes para conseguir el costo más bajo, o como `Transactional`, que optimiza el envío de mensajes para conseguir la máxima fiabilidad.

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Establecimiento de atributos de SMS

Para actualizar la configuración predeterminada de los mensajes SMS, utilice la operación [SetSMSAttributes](#).

Este ejemplo establece el atributo `DefaultSMSType` en `Transactional`, que optimiza el envío de mensajes para conseguir la máxima fiabilidad.

Importaciones

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



```
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Comprobación de si se ha desactivado un número de teléfono

Para determinar si un propietario de número de teléfono concreto ha desactivado la recepción de mensajes SMS desde su cuenta, utilice la operación [CheckIfPhoneNumberIsOptedOut](#).

En este ejemplo, el número de teléfono está en formato E.164, un estándar utilizado para las telecomunicaciones internacionales.

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Lista de los números de teléfono desactivados

Para recuperar una lista de números de teléfono en los que el propietario ha desactivado la recepción de mensajes SMS desde su cuenta, utilice la operación [ListPhoneNumbersOptedOut](#).

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Publicación en un mensaje de texto (mensaje SMS)

Para entregar un mensaje de texto (mensaje SMS) directamente a un número de teléfono, utilice la operación [Publish](#).

En este ejemplo, el número de teléfono está en formato E.164, un estándar utilizado para las telecomunicaciones internacionales.

Los mensajes SMS puede contener hasta 140 bytes. El límite de tamaño de una sola acción de publicación SMS es de 1600 bytes.

Para obtener más información sobre cómo enviar mensajes SMS, consulte [Envío de un mensaje SMS](#).

Importaciones

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

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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 = $SqsClient->publish([  
        'Message' => $message,  
        'PhoneNumber' => $phone,  
    ]);  
    var_dump($result);  
} catch (AwsException $e) {  
    // output error message if fails  
    error_log($e->getMessage());  
}
```

Ejemplos de Amazon SQS que utilizan el AWS SDK for PHP versión 3

Amazon Simple Queue Service (SQS) es un servicio de cola de mensajes rápido, fiable, escalable y totalmente administrado. Amazon SQS permite desacoplar los componentes de una aplicación en la nube. Amazon SQS incluye colas estándar con un alto rendimiento y procesamiento "al menos una vez", y colas FIFO que ofrecen distribución FIFO (primero en entrar, primero en salir) y procesamiento "exactamente una vez".



Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Temas

- [Habilitación del sondeo largo en Amazon SQS con el AWS SDK for PHP versión 3 \(p. 292\)](#)
- [Administración del tiempo de espera de visibilidad en Amazon SQS con el AWS SDK for PHP versión 3 \(p. 295\)](#)
- [Envío y recepción de mensajes en Amazon SQS con el AWS SDK for PHP versión 3 \(p. 296\)](#)
- [Uso de colas de mensajes fallidos en Amazon SQS con el AWS SDK for PHP versión 3 \(p. 298\)](#)
- [Uso de colas en Amazon SQS con el AWS SDK para PHP versión 3 \(p. 299\)](#)

Habilitación del sondeo largo en Amazon SQS con el AWS SDK for PHP versión 3

El sondeo largo reduce la cantidad de respuestas vacías al permitir que Amazon SQS espere durante un tiempo especificado hasta que haya un mensaje disponible en la cola antes de enviar una respuesta. Asimismo, el sondeo largo elimina las respuestas vacías falsas, ya que consulta todos los servidores en lugar de solo una muestra de servidores. Para habilitar el sondeo largo, especifique un tiempo de espera diferente de cero para los mensajes recibidos. Para obtener más información, consulte [Sondeo largo de Amazon SQS](#).

Los siguientes ejemplos muestran cómo:

- Establecer atributos en una cola de Amazon SQS para habilitar el sondeo largo utilizando [SetQueueAttributes](#).
- Recuperar uno o varios mensajes con sondeo largo utilizando [ReceiveMessage](#).
- Creación de una cola de sondeo largo mediante [CreateQueue](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Establecer atributos en una cola para habilitar el sondeo largo

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Recuperar mensajes con el sondeo largo

Importaciones

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

use Aws\Sqs\SqsClient;
```

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

Código de muestra

```
$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());
}
```

Crear una cola con el sondeo largo

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Administración del tiempo de espera de visibilidad en Amazon SQS con el AWS SDK for PHP versión 3

Un tiempo de espera de visibilidad es un intervalo durante el cual Amazon SQS impide que otros componentes consumidores reciban y procesen un mensaje. Para obtener más información, consulte [Tiempo de espera de visibilidad](#).

El siguiente ejemplo muestra cómo:

- cambiar el tiempo de espera de visibilidad de los mensajes especificados en una cola por nuevos valores, utilizando [ChangeMessageVisibilityBatch](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Cambiar el tiempo de espera de visibilidad de varios mensajes

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Envío y recepción de mensajes en Amazon SQS con el AWS SDK for PHP versión 3

Para obtener más información acerca de los mensajes de Amazon SQS, consulte [Envío de un mensaje a una cola de SQS](#) y [Recepción y eliminación de un mensaje de una cola de SQS](#) en la Amazon SQS Developer Guide.

Los siguientes ejemplos muestran cómo:

- Entregue un mensaje en una cola especificada mediante [SendMessage](#).
- Recupere uno o varios mensajes (hasta 10) de una cola especificada con [ReceiveMessage](#).
- Elimine un mensaje de una cola mediante [DeleteMessage](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3](#) (p. 7).

Enviar un mensaje

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Recibir y eliminar mensajes

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ));
    if (!empty($result->get('Messages'))) {
        var_dump($result->get('Messages')[0]);
        $result = $client->deleteMessage([
```



```
'QueueUrl' => $queueUrl, // REQUIRED
'ReceiptHandle' => $result->get('Messages')[0]['ReceiptHandle'] // REQUIRED
]);
} else {
    echo "No messages in queue. \n";
}
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Uso de colas de mensajes fallidos en Amazon SQS con el AWS SDK for PHP versión 3

Una cola de mensajes fallidos es una cola a la que otras colas (de origen) pueden enviar mensajes que no se han podido procesar correctamente. Puede apartar y aislar estos mensajes en la cola de mensajes fallidos para determinar por qué no se procesaron correctamente. Debe configurar individualmente cada cola de origen que envía mensajes a una cola de mensajes fallidos. Varias colas pueden dirigirse a una única cola de mensajes fallidos.

Para obtener más información, consulte [Uso de colas de mensajes fallidos de SQS](#).

El siguiente ejemplo muestra cómo:

- Habilite una cola de mensajes fallidos mediante [SetQueueAttributes](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Habilitar una cola de mensajes fallidos

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Uso de colas en Amazon SQS con el AWS SDK para PHP versión 3

Para obtener más información acerca de las colas de Amazon SQS, consulte [Cómo funcionan las colas de Amazon SQS](#).

Los siguientes ejemplos muestran cómo:

- Devolver una lista de sus colas utilizando [ListQueues](#).
- Creación de una cola nueva con [CreateQueue](#).
- Devolver la URL de una cola existente utilizando [GetQueueUrl](#).
- Elimine una cola especificada mediante [DeleteQueue](#).

Todo el código de ejemplo de AWS SDK for PHP versión 3 está disponible [aquí en GitHub](#).

Credentials

Antes de ejecutar el código de ejemplo, configure sus credenciales de AWS, tal como se describe en [Credenciales para el AWS SDK para PHP versión 3 \(p. 44\)](#). A continuación, importe el AWS SDK for PHP, como se describe en [Uso básico del AWS SDK para PHP versión 3 \(p. 7\)](#).

Devolver una lista de colas

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sqs\SqsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Creación de una cola

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sqs\SqsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());  
}
```

Devolver la URL de una cola

Importaciones

```
require 'vendor/autoload.php';  
  
use Aws\Sqs\SqsClient;  
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Eliminar una cola

Importaciones

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

use Aws\Sqs\SqsClient;
use Aws\Exception\AwsException;
```

Código de muestra

```
$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());
}
```

Seguridad de this AWS Product or Service

La seguridad en la nube de Amazon Web Services (AWS) es lo más importante para nosotros. Como cliente de AWS, se beneficiará de una arquitectura de red y un centro de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre AWS y usted. En el [modelo de responsabilidad compartida](#), se habla de «seguridad de la nube» y «seguridad en la nube»:

Seguridad de la nube: AWS es responsable de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la nube de AWS y de proporcionar servicios que puedan utilizarse de forma segura. En AWS, nuestra mayor prioridad es nuestra responsabilidad en materia de seguridad, y hay auditores externos que comprueban y verifican con frecuencia la eficacia de nuestra seguridad en los [programas de conformidad de AWS](#).

Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que usa y otros factores, como la confidencialidad de los datos, los requisitos de la organización y la legislación y reglamentos aplicables.

Temas

- [Protección de datos en this AWS Product or Service \(p. 302\)](#)
- [Identity and Access Management en this AWS Product or Service \(p. 303\)](#)
- [Validación de la conformidad para this AWS Product or Service \(p. 303\)](#)
- [Resiliencia para this AWS Product or Service \(p. 304\)](#)
- [Seguridad de la infraestructura para this AWS Product or Service \(p. 305\)](#)
- [Migración de cliente de cifrado de Amazon S3 \(p. 305\)](#)

Protección de datos en this AWS Product or Service

El [modelo de responsabilidad compartida](#) se aplica a la protección de datos en este producto o servicio de AWS. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye las tareas de configuración y administración de seguridad para los servicios de AWS que utilice. Para obtener más información sobre la privacidad de datos, consulte las [preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte la entrada de blog relativa al [modelo de responsabilidad compartida de AWS y GDPR](#) en el blog de seguridad de AWS.

Para fines de protección de datos, le recomendamos que proteja las credenciales de la cuenta de AWS y configure cuentas de usuario individuales con AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir con sus obligaciones laborales. También le recomendamos proteger sus datos de las siguientes formas:

- Utilice la autenticación multifactor (MFA) con cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. Le recomendamos TLS 1.2 o una versión posterior.

- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Use soluciones de cifrado de AWS con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- Utilice los servicios de seguridad administrados avanzados como, por ejemplo, Amazon Macie, que ayudan a detectar y proteger los datos personales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de línea de comandos o una API, utilice un punto de enlace de FIPS. Para obtener más información acerca de los puntos de enlace de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 140-2](#).

Le recomendamos encarecidamente que nunca incluya información de identificación confidencial (por ejemplo, números de cuenta de sus clientes) en los campos de formato libre, como el campo Name (Nombre). Esto incluye cuando trabaje con this AWS product or service u otros servicios de AWS a través de la consola, la API, la AWS CLI o AWS SDKs. Cualquier dato que escriba en this AWS product or service o en otros servicios se puede incluir en los registros de diagnóstico. Cuando proporcione una URL a un servidor externo, no incluya información sobre las credenciales en la URL para validar la solicitud en ese servidor.

Identity and Access Management en this AWS Product or Service

AWS Identity and Access Management (IAM) es un servicio de Amazon Web Services (AWS) que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién obtiene autorización (tiene permiso) para utilizar los recursos de los servicios de AWS. IAM es un servicio de AWS que puede utilizarse sin ningún costo adicional.

Si desea usar this AWS product or service para acceder a AWS, necesita una cuenta de AWS y unas credenciales de AWS. Para aumentar la seguridad de su cuenta de AWS, le recomendamos que utilice un usuario de IAM a la hora de proporcionar credenciales de acceso en lugar de usar las credenciales de su cuenta de AWS.

Para obtener más información acerca de cómo trabajar con IAM, consulte [IAM](#).

Para obtener información general sobre los usuarios de IAM y por qué son importantes para la seguridad de su cuenta, consulte [Credenciales de seguridad de AWS](#) en la [Referencia general de Amazon Web Services](#).

This AWS product or service cumple el [modelo de responsabilidad compartida](#) en los servicios concretos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de cada programa](#).

Validación de la conformidad para this AWS Product or Service

This AWS product or service cumple el [modelo de responsabilidad compartida](#) en los servicios concretos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios

de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de cada programa](#).

Hay auditores externos que evalúan la seguridad y la conformidad de los servicios de Amazon Web Services (AWS) en diferentes programas de conformidad de AWS. Entre estos se incluyen SOC, PCI, FedRAMP, HIPAA y otros. AWS proporciona una lista actualizada frecuentemente de los servicios de AWS que se encuentran en el ámbito de programas de conformidad específicos en [Servicios de AWS en el ámbito del programa de conformidad](#).

Los informes de auditoría de terceros están disponibles para su descarga utilizando AWS Artifact. Para obtener más información, consulte la sección [Descarga de informes en AWS Artifact](#).

Para obtener más información acerca de los programas de conformidad de AWS, consulte [Programas de conformidad de AWS](#).

Su responsabilidad respecto a la conformidad al utilizar this AWS product or service para acceder a un servicio de AWS se determina en función de la confidencialidad de los datos y los objetivos de conformidad de su organización, así como la legislación y los reglamentos aplicables. El uso de un servicio de AWS está sujeto a conformidad con normas tales como HIPAA, PCI o FedRAMP, AWS proporciona recursos para ayudarle:

- [Guías de inicio rápido de seguridad y conformidad](#): guías de implementación que contienen consideraciones sobre la arquitectura y ofrecen pasos para implementar entornos de referencia centrados en la seguridad y la conformidad de AWS.
- [Documento técnico sobre el diseño de arquitecturas para cumplir los requisitos de seguridad y conformidad de HIPAA](#): documento técnico en el que se describe cómo las empresas pueden utilizar AWS para crear aplicaciones conformes con HIPAA.
- [AWS Compliance Resources](#): un conjunto de manuales y guías que podría aplicarse a su sector y ubicación.–
- [AWS Config](#) un servicio que evalúa en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y las normativas.–
- [AWS Security Hub](#): servicio que le ofrece una panorámica integral de su estado de seguridad en AWS, lo que le ayuda a comprobar la conformidad con las prácticas recomendadas y las normas de seguridad sectoriales.

Resiliencia para this AWS Product or Service

La infraestructura global de Amazon Web Services (AWS) se compone de regiones y zonas de disponibilidad de AWS.

Las regiones de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia.

Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad y las regiones de AWS, consulte [Infraestructura global de AWS](#).

This AWS product or service cumple el [modelo de responsabilidad compartida](#) en los servicios concretos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de cada programa](#).

Seguridad de la infraestructura para this AWS Product or Service

This AWS product or service cumple el [modelo de responsabilidad compartida](#) en los servicios concretos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de cada programa](#).

Para obtener información sobre los procesos de seguridad de AWS, consulte [AWS: Documento técnico Información general sobre procesos de seguridad](#).

Migración de cliente de cifrado de Amazon S3

En este tema se muestra cómo migrar las aplicaciones de la versión 1 (V1) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) a la versión 2 (V2) y garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración.

Información general sobre la migración

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. En primer lugar, implemente una versión actualizada del AWS SDK for PHP en su aplicación. Esto permite que los clientes de cifrado V1 existentes descifren objetos escritos por los nuevos clientes V2. Si su aplicación utiliza varias SDKs de AWS, debe actualizar cada SDK por separado.
2. Se migran los clientes de cifrado y descifrado a V2. Cuando todos los clientes de cifrado V1 puedan leer nuevos formatos, podrá migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones V2.

Actualizar clientes existentes para leer nuevos formatos

El cliente de cifrado V2 utiliza algoritmos de cifrado que las versiones anteriores del cliente no admiten. El primer paso en la migración consiste en actualizar sus clientes de descifrado V1 a la última versión del SDK. Tras completar este paso, los clientes V1 de su aplicación podrán descifrar objetos cifrados por los clientes de cifrado V2. Consulte los detalles a continuación para cada versión principal de la AWS SDK for PHP.

Actualización del AWS SDK para PHP versión 3

La versión 3 es la última versión de la AWS SDK for PHP. Para completar esta migración, debe utilizar la versión 3.148.0 o posterior del paquete `aws/aws-sdk-php`.

Instalación desde la línea de comandos

Para los proyectos que se instalaron con Composer, en el archivo `Composer`, actualice el paquete del SDK a la versión 3.148.0 del SDK y, a continuación, ejecute el siguiente comando.

```
composer update aws/aws-sdk-php
```


Instalación mediante el archivo phar o zip

Utilice uno de los siguientes métodos. Asegúrese de colocar el archivo del SDK actualizado en la ubicación requerida por el código, lo cual viene determinado por la instrucción require.

Para los proyectos que se instalaron con el archivo Phar, obtenga el archivo actualizado de <https://docs.aws.amazon.com/aws-sdk-php/v3/download/aws.phar>.

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

Para los proyectos que se instalaron con el archivo Zip, obtenga el archivo actualizado de <https://docs.aws.amazon.com/aws-sdk-php/v3/download/aws.zip>.

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

Migración de clientes de cifrado y descifrado a V2

Después de actualizar sus clientes para leer los nuevos formatos de cifrado, puede actualizar sus aplicaciones a los clientes de cifrado y descifrado V2. Los pasos siguientes muestran cómo migrar correctamente el código de V1 a V2.

Requisitos para la actualización a clientes V2

1. El contexto de cifrado de AWS KMS debe pasarse a los métodos `S3EncryptionClientV2::putObject` y `S3EncryptionClientV2::putObjectAsync`. El contexto de cifrado de AWS KMS es una matriz asociativa de pares clave-valor, que debe añadir al contexto de cifrado para el cifrado de claves de AWS KMS. Si no se requiere contexto adicional, puede pasar una matriz vacía.
2. El `@SecurityProfile` debe pasarse a los métodos `getObject` y `getObjectAsync` en `S3EncryptionClientV2`. `@SecurityProfile` es un nuevo parámetro obligatorio de los métodos `getObject...`. Si se establece en `'V2'`, solo se pueden descifrar los objetos cifrados en formato compatible con V2. El establecimiento de este parámetro en `'V2_AND_LEGACY'` también permite descifrar los objetos cifrados en formato compatible con V1. Para admitir la migración, establezca `@SecurityProfile` en `'V2_AND_LEGACY'`. Utilice `'V2'` solo para el desarrollo de nuevas aplicaciones.
3. (opcional) Incluya el parámetro `@KmsAllowDecryptWithAnyCmk` en `S3EncryptionClientV2::getObject` y `S3EncryptionClientV2::getObjectAsync` methods.. Se ha agregado un nuevo parámetro llamado `@KmsAllowDecryptWithAnyCmk`. El establecimiento de este parámetro en `true` habilita el descifrado sin proporcionar una clave de KMS. El valor predeterminado es `false`.
4. Para el descifrado con un cliente V2, si el parámetro `@KmsAllowDecryptWithAnyCmk` no está establecido en `true` para las llamadas al método `"getObject..."`, debe suministrarse `kms-key-id` al constructor `KmsMaterialsProviderV2`.

Ejemplos de migración

Ejemplo 1: Migración a clientes V2

Migración previa

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Con posterioridad a la migración

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Ejemplo 2: Uso de KMS con kms-key-id

Note

Estos ejemplos utilizan importaciones y variables definidas en el Ejemplo 1. Por ejemplo, `$encryptionClient`.

Migración previa

```
use Aws\Crypto\KmsMaterialsProvider;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProvider(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

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

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

$result = $encryptionClient->getObject([
```

```
'@MaterialsProvider' => $materialsProvider,  
'@CipherOptions' => $cipherOptions,  
'Bucket' => $bucket,  
'Key' => $key,  
]);
```

Con posterioridad a la migración

```
use Aws\Crypto\KmsMaterialsProviderV2;  
use Aws\Kms\KmsClient;  
  
$kmsKeyId = 'kms-key-id';  
$materialsProvider = new KmsMaterialsProviderV2(  
    new KmsClient([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => 'latest',  
    ]),  
    $kmsKeyId  
);  
  
$bucket = 'the-bucket-name';  
$key = 'the-file-name';  
$cipherOptions = [  
    'Cipher' => 'gcm',  
    'KeySize' => 256,  
];  
  
$encryptionClient->putObject([  
    '@MaterialsProvider' => $materialsProvider,  
    '@CipherOptions' => $cipherOptions,  
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],  
    'Bucket' => $bucket,  
    'Key' => $key,  
    'Body' => fopen('file-to-encrypt.txt', 'r'),  
]);  
$result = $encryptionClient->getObject([  
    '@KmsAllowDecryptWithAnyCmk' => true,  
    '@SecurityProfile' => 'V2_AND_LEGACY',  
    '@MaterialsProvider' => $materialsProvider,  
    '@CipherOptions' => $cipherOptions,  
    'Bucket' => $bucket,  
    'Key' => $key,  
]);
```

Preguntas frecuentes sobre AWS SDK for PHP versión 3

¿Qué métodos están disponibles en un cliente?

AWS SDK for PHP utiliza las descripciones de servicio y los [métodos magic __call\(\)](#) dinámicos para ejecutar las operaciones de la API. Puede encontrar una lista completa de métodos disponibles para un cliente de servicio web en la [documentación de la API](#) del cliente.

¿Qué debo hacer si aparece un error de certificado SSL de cURL?

Este problema puede ocurrir cuando se usa un paquete de CA desactualizado con cURL y SSL. Puede solucionar este problema actualizando el paquete de CA en su servidor o descargando un paquete de CA actualizado desde el [sitio web de cURL directamente](#).

De forma predeterminada, AWS SDK for PHP utilizará el paquete de CA que se configura al compilar PHP. Puede cambiar el paquete de CA predeterminado que utiliza PHP modificando el valor de configuración del archivo .ini de PHP `openssl.cafile` que se va a establecer en la ruta de un archivo de CA en el disco.

¿Qué versiones de la API están disponibles para un cliente?

Disponer de una opción `version` es obligatorio a la hora de crear un cliente. Puede encontrar una lista de las versiones de la API disponibles en la página la documentación de la API de cada cliente `::aws-php-class:<index.html>`. Si no consigue cargar una versión específica de la API, es posible que tenga que actualizar su copia de AWS SDK for PHP.

Puede proporcionar la cadena `latest` con el valor de configuración "version" para utilizar la versión de la API más reciente disponible que encuentre su proveedor de API del cliente (el `api_provider` predeterminado escaneará el directorio `src/data` del SDK en busca de modelos de la API).

Warning

No le recomendamos utilizar `latest` en una aplicación de producción, ya que integrar una nueva versión secundaria del SDK que incluya una actualización de la API podría romper su aplicación de producción.

¿Qué versiones de Region están disponibles para un cliente?

Disponer de una opción `region` es obligatorio a la hora de crear un cliente y se especifica mediante un valor de cadena. Para ver una lista de las regiones y puntos de enlace de AWS disponibles, consulte [Regiones y puntos de enlace de AWS](#) en la Amazon Web Services General Reference.

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

¿Por qué no puedo cargar o descargar archivos de más de 2 GB?

Dado que el tipo de número entero de PHP está firmado y muchas plataformas utilizan números enteros de 32 bits, AWS SDK for PHP no gestiona correctamente los archivos de más de 2 GB en una pila de 32 bits (bajo "pila" se engloba la CPU, el SO, el servidor web y el binario de PHP). Se trata de un [problema de PHP conocido](#). En el caso de Microsoft Windows, solo crea enteros de 64 bits que admiten PHP 7.

La solución recomendada es utilizar una [pila de Linux de 64 bits](#), como la AMI de Amazon Linux de 64 bits, con la versión más reciente de PHP instalada.

Para obtener más información, consulte [PHP filesize: : valores devueltos](#).

¿Cómo puedo ver qué datos se envían a través de la red?

Puede obtener información de depuración, incluidos los datos enviados a través de la red, utilizando la opción `debug` de un constructor de clientes. Cuando esta opción se establece en `true`, todas las mutaciones del comando que se esté ejecutando, la solicitud que se envíe, la respuesta que se reciba y el resultado que se esté procesando se emiten a STDOUT. Esto incluye los datos que se envían y reciben a través de la red.

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

¿Cómo puedo establecer los encabezados arbitrarios en una solicitud?

Puede añadir cualquier encabezado arbitrario a una operación de servicio añadiendo un middleware personalizado a `Aws\HandlerList` de una `Aws\CommandInterface` o `Aws\ClientInterface`. El siguiente ejemplo muestra cómo añadir un encabezado `X-Foo-Baz` a una operación de Amazon S3 `PutObject` específica utilizando el método auxiliar `Aws\Middleware::mapRequest`.

Para obtener más información, consulte [mapRequest \(p. 76\)](#).

¿Cómo puedo firmar una solicitud arbitraria?

Puede firmar una :aws-php-class arbitraria: del PSR-7 de <class-Psr.Http.Message.RequestInterface.html> con la clase :aws-php-class del SDK: SignatureV4 clase <class-Aws.Signature.SignatureV4.html>.

Para obtener un ejemplo completo de cómo hacer esto, consulte [Firma de solicitudes personalizadas del dominio de Amazon CloudSearch con el AWS SDK para PHP versión 3](#) (p. 142).

¿Cómo puedo modificar un comando antes de enviarlo?

Puede modificar un comando antes de enviarlo añadiendo un middleware personalizado a `Aws\HandlerList` de una `Aws\CommandInterface` o `Aws\ClientInterface`. El siguiente ejemplo muestra cómo añadir parámetros de comando personalizados a un comando antes de que se envíe, básicamente añadiendo opciones predeterminadas. En este ejemplo se utiliza el método de ayuda `Aws\Middleware::mapCommand`.

Para obtener más información, consulte [mapCommand](#) (p. 75).

¿Qué es un `CredentialsException`?

Si visualiza una excepción `Aws\Exception\CredentialsException` al utilizar AWS SDK for PHP, significa que el SDK se ha proporcionado sin credenciales y no ha podido encontrar credenciales en el entorno.

Si crea una instancia de un cliente sin credenciales, la primera vez que realice una operación de servicio el SDK intentará encontrar credenciales. Primero comprueba algunas variables de entorno específicas y, a continuación, busca las credenciales del perfil de instancia, que solo están disponibles en instancias Amazon EC2 configuradas. Si no se proporcionan o encuentran credenciales, se lanza una `Aws\Exception\CredentialsException`.

Si ve este error y quiere utilizar las credenciales del perfil de instancia, debe asegurarse de que la instancia Amazon EC2 en la que se está ejecutando el SDK se configure con un rol de IAM adecuado.

Si ve este error y no tiene previsto utilizar las credenciales del perfil de instancia, tiene que asegurarse de que está proporcionando las credenciales correctamente al SDK.

Para obtener más información, consulte [Credenciales para el AWS SDK para PHP versión 3](#) (p. 44).

¿AWS SDK for PHP se puede utilizar en HHVM?

Actualmente, AWS SDK for PHP no se ejecuta en HHVM y no se podrá ejecutar hasta que se resuelva el [problema de la semántica de rendimiento en HHVM](#).

¿Cómo se deshabilita SSL?

Puede deshabilitar SSL estableciendo el parámetro `scheme` en un método de fábrica de cliente en "http". Es importante tener en cuenta que no todos los servicios admiten el acceso http. Para ver una lista de las

regiones, puntos de enlace y esquemas admitidos, consulte [Regiones y puntos de enlace](#) en la Amazon Web Services General Reference.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

Warning

Puesto que SSL requiere que se cifren todos los datos y necesita más paquetes TCP para completar un protocolo de conexión que únicamente TCP, es posible que si deshabilita SSL mejore ligeramente el rendimiento. Sin embargo, con SSL deshabilitado, todos los datos se envían cifrados a través de la red. Antes de deshabilitar SSL, debe considerar cuidadosamente las implicaciones de seguridad y la posibilidad de que se produzca un acceso no autorizado a través de la red.

¿Qué debo hacer cuando aparece un "Error de análisis"?

El motor PHP lanzará errores de análisis cuando encuentre sintaxis que no entiende. Esto suele ocurrir cuando intenta ejecutar código que se ha creado para una versión diferente de PHP.

Si se produce un error de análisis, compruebe el sistema y asegúrese de que cumple los [Requisitos y recomendaciones para el AWS SDK para PHP versión 3 del SDK](#) (p. 4).

¿Por qué el cliente de Amazon S3 descomprime los archivos gz?

Algunos controladores HTTP, incluido el controlador HTTP Guzzle 6 predeterminado, aumentará los cuerpos de respuesta comprimidos de forma predeterminada. Puede anular este comportamiento estableciendo la opción HTTP [decode_content](#) (p. 33) en `false`. Por motivos de compatibilidad con versiones anteriores, este valor predeterminado no se puede cambiar, pero le recomendamos que deshabilite la decodificación de contenido en el nivel de cliente de S3.

Consulte [decode_content](#) (p. 33) para ver un ejemplo de cómo deshabilitar la decodificación automática de contenido.

¿Cómo puedo deshabilitar la firma de cuerpos en Amazon S3?

Puede deshabilitar la firma de cuerpos estableciendo el parámetro `ContentSHA256` en el objeto de comandos en `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`. A continuación, AWS SDK for PHP lo utilizará como el encabezado `x-amz-content-sha-256` y la suma de comprobación del cuerpo en la solicitud canónica.

```
$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);
```

¿Cómo se gestiona el esquema de reintento en AWS SDK for PHP?

AWS SDK for PHP tiene un `RetryMiddleware` que gestiona el comportamiento de reintento. En términos de códigos de estado HTTP de 5xx para errores de servidor, el SDK reintenta en 500, 502, 503 y 504.

Las excepciones de limitación controlada, incluidas `RequestLimitExceeded`, `Throttling`, `ProvisionedThroughputExceededException`, `ThrottlingException`, `RequestThrottled` y `BandwidthLimitExceeded`, también se gestionan con reintentos.

AWS SDK for PHP también integra el retraso exponencial con un algoritmo de fluctuación y retardo en el esquema de reintento. Además, el comportamiento de reintento predeterminado se configura como 3 para todos los servicios, excepto Amazon DynamoDB, que es 10.

¿Cómo se gestionan las excepciones con códigos de error?

Además de las clases AWS SDK for PHP personalizadas para `Exception`, cada servicio de AWS tiene su propia clase de excepción que hereda de [AwsException](#). Puede determinar más tipos de error específicos para detectar con los errores específicos de la API que se enumeran en la sección `Errors` de cada método.

La información del código de error está disponible en [getAwsErrorCode\(\)](#) desde `Aws\Exception\AwsException`.

```
$sns = new \Aws\Sns\SnsClient([
    'region' => 'us-west-2',
    'version' => 'latest',
]);

try {
    $sns->publish([
        // parameters
        ...
    ]);
    // Do something
} catch (\AwsException $e) {
```



```
switch ($e->getAwsErrorCode()) {  
    case 'EndpointDisabled':  
    case 'NotFound':  
        // Do something  
        break;  
    }  
}
```

Glossary

Versión de API

Los servicios tienen una o varias versiones de la API, y la versión que se usa determina qué operaciones y parámetros son válidos. La versión de la API se indican con el formato de una fecha. Por ejemplo, la última versión de la API para Amazon S3 es 2006-03-01. [Especifique una versión \(p. 44\)](#) al configurar un objeto de cliente.

Cliente

Los objetos cliente se usan para ejecutar operaciones para un servicio. Cada servicio contemplado en el SDK tiene un objeto de cliente correspondiente. Los objetos cliente tienen métodos que se corresponden directamente con las operaciones del servicio. Consulte la [guía de uso básico \(p. 7\)](#) para obtener más información acerca de cómo crear y utilizar objetos de cliente.

Comando

Los objetos comando encapsulan la ejecución de una operación. Si sigue los [patrones de uso básicos \(p. 7\)](#) del SDK, no tratará directamente con objetos comando. Para el acceso a los objetos comando puede usarse el método `getCommand()` de un cliente y así poder usar características avanzadas del SDK, como solicitudes simultáneas y procesamiento por lotes. Para obtener información detallada, consulte [Objetos comando en el AWS SDK para PHP versión 3 \(p. 61\)](#).

Credentials

Para interactuar con los servicios de AWS, auténtíquese en el servicio con sus credenciales o sus [claves de acceso de AWS](#). Las claves de acceso constan de dos partes: su ID de clave de acceso, que identifica su cuenta, y su clave de acceso secreta, que se utiliza para crear firmas cuando se ejecutan operaciones. [Proporcione las credenciales \(p. 44\)](#) cuando configure un objeto de cliente.

Controlador

Un controlador es una función que realiza la transformación real de un comando y una solicitud en un resultado. Un controlador suele enviar solicitudes HTTP. Los controladores pueden incluir middleware para aumentar su comportamiento. Un controlador es una función que acepta una `Aws\CommandInterface` y una `Psr\Http\Message\RequestInterface` y devuelve una promesa que se cumple con una `Aws\ResultInterface` o se rechaza con un motivo `Aws\Exception\AwsException`.

JMESPath

[JMESPath](#) es un lenguaje de consultas para datos de tipo JSON. El AWS SDK for PHP utiliza expresiones de JMESPath para consultar estructuras de datos PHP. : las expresiones JMESPath de `Aws\Result` se pueden utilizar directamente en objetos `Aws\ResultPaginator` y `search($expression)` a través del método .

Middleware

El middleware es un tipo especial de función de alto nivel que aumenta el comportamiento de la transferencia de un comando y lo delega al "siguiente" controlador. Las funciones de middleware aceptan una `Aws\CommandInterface` y una `Psr\Http\Message\RequestInterface` y devuelven una promesa que se cumple con una `Aws\ResultInterface` o se rechaza con un motivo `Aws\Exception\AwsException`.

Operation

Se refiere a una sola operación dentro de la API de un servicio (por ejemplo, `CreateTable` para DynamoDB o `RunInstances` para Amazon EC2). En el SDK, las operaciones se ejecutan llamando al método con el mismo nombre del objeto cliente del servicio correspondiente. La ejecución de

una operación implica la preparación y el envío de una solicitud HTTP al servicio y el análisis de la respuesta. El SDK abstrae este proceso de ejecutar una operación mediante objetos comando.

Paginador

Algunas operaciones de servicios de AWS están paginadas y responden con resultados truncados. Por ejemplo, la operación Amazon S3 de `ListObjects` devuelve un máximo de 1000 objetos cada vez. Las operaciones como estas requieren realizar solicitudes posteriores con parámetros de token (o marcador) para recuperar todo el conjunto de resultados. Los paginadores son una característica del SDK que actúan como una abstracción sobre este proceso para facilitar a los desarrolladores el uso de APIs paginado. : se obtiene acceso a ellos a través del método `getPaginator()` del cliente. Para obtener información detallada, consulte [Paginadores en el AWS SDK para PHP versión 3 \(p. 82\)](#).

Promesa

Una promesa representa el resultado final de una operación asíncrona. La forma principal de interactuar con una promesa es a través de su método "then", que registra devoluciones de llamada para recibir el valor final de la promesa o la razón por la que no puede cumplirse.

Región

Los servicios están disponibles en [una o varias regiones geográficas](#). Los servicios pueden tener diferentes puntos de enlace o URL en cada región a fin de reducir la latencia de datos de las aplicaciones. [Proporcione una región \(p. 39\)](#) al configurar un objeto de cliente para que el SDK pueda determinar el punto de enlace que se debe utilizar con el servicio.

SDK

El término "SDK" puede referirse a la biblioteca AWS SDK for PHP en su conjunto, pero también a la clase `Aws\Sdk` ([docs](#)), que actúa como una fábrica para los objetos cliente de cada servicio. La clase `Sdk` también permite indicar un conjunto de [valores de configuración globales \(p. 24\)](#) que se aplican a todos los objetos cliente que crea.

Servicio

que es una forma general de referirse a cualquiera de los servicios de AWS (por ejemplo, Amazon S3, Amazon DynamoDB, AWS OpsWorks, etc.). Cada servicio tiene un objeto de cliente correspondiente en el SDK que es compatible con una o varias versiones de la API. Cada servicio también tiene una o varias operaciones que componen su API. Los servicios están disponibles en una o más regiones.

Signature (Firma)

Cuando ejecutan operaciones, el SDK utiliza sus credenciales para crear una firma digital de la solicitud. El servicio verifica entonces la firma antes de procesar su solicitud. El SDK encapsula este proceso de firma, que se sigue de forma automática usando las credenciales que ha configurado para el cliente.

Esperador

Los esperadores son una característica del SDK que facilitan el trabajo con las operaciones que cambian el estado de un recurso y que presentan consistencia final o son de naturaleza asíncrona. Por ejemplo, la operación `CreateTable` de Amazon DynamoDB envía una respuesta inmediata, pero la tabla puede no estar lista para el acceso hasta que transcurran varios segundos. El uso de un esperador permite esperar hasta que un recurso alcance un estado determinado, manteniéndose inactivo y sondeando el estado del recurso. El acceso a los esperadores se hace con el método `waitUntil()` del cliente. Para obtener información detallada, consulte [Esperadores en el AWS SDK para PHP versión 3 \(p. 84\)](#).

Para ver la terminología más reciente de AWS, consulte [Glosario de AWS](#) en la Referencia general de AWS.

Recursos de adicionales

Los siguientes enlaces proporcionan recursos adicionales que puede usar con el SDK para PHP versión 3.

Foro del SDK para PHP

Encontrará preguntas y debates sobre temas de interés para los usuarios del SDK para PHP en el [Foro de SDK para PHP](#).

PHP SDK v3 y guía para desarrolladores en GitHub

Hay varios repositorios en GitHub para el AWS SDK for PHP.

- El AWS SDK for PHP actual está disponible en el [repositorio del SDK](#).
- La guía para desarrolladores del SDK para PHP está disponible en formato de texto reestructurado en su propio [repositorio de la documentación](#).
- El código de muestra incluido en esta guía está disponible en el [repositorio del código de muestra del SDK](#).

SDK para PHP en Gitter

También puede encontrar preguntas y debates acerca del AWS SDK for PHP en la [comunidad del SDK para PHP](#) en Gitter.

Historial del documento

En la siguiente tabla se describen los cambios importantes que se han realizado en la AWS SDK for PHP Developer Guide desde la última versión.

Última actualización importante de la documentación: 7 de agosto de 2020

Cambiar	Descripción	Fecha de la versión
Cifrado de Amazon S3	Adición de un tema sobre la migración del cliente de cifrado de Amazon S3	7 de agosto de 2020
Ejemplos de Secrets Manager	Adición de más ejemplos de servicios	27 de marzo de 2019
Detección de puntos de enlace	Configuración de la detección de puntos de enlace	15 de febrero de 2019
Amazon CloudFront	Adición de más ejemplos de servicios	25 de enero de 2019
Características de los servicios	Métricas de SDK	11 de enero de 2018
Amazon Kinesis, Amazon SNS	Adición de más ejemplos de servicios	14 de diciembre de 2018
Ejemplos de Amazon SES	Adición de más ejemplos de servicios	5 de octubre de 2018
Ejemplos de AWS KMS	Adición de más ejemplos de servicios	8 de agosto de 2018
Credentials	Aclaración y simplificación de la guía de credenciales	30 de junio de 2018
Ejemplos de MediaConvert	Adición de más ejemplos de servicios	15 de junio de 2018
Nuevo diseño web	Cambio de la documentación al estilo de AWS	9 de mayo de 2018
Cifrado de Amazon S3	Cifrado del lado del cliente	17 de noviembre de 2017
Amazon S3, Amazon SQS	Adición de más ejemplos de servicios	26 de marzo de 2017
Amazon S3, IAM, Amazon EC2	Adición de más ejemplos de servicios	17 de marzo de 2017
Adición de credenciales	Agrega compatibilidad para AssumeRole e ini	17 de enero de 2017
Ejemplos de S3	S3 para varias regiones y publicaciones prefiradas	18 de marzo de 2016

Cambiar	Descripción	Fecha de la versión
Amazon ES y Amazon CloudSearch	Adición de más ejemplos de servicios	28 de diciembre de 2015
Línea de comandos	Adición de parámetros de comandos	13 de agosto de 2015
Características de los servicios	Adición de características de los servicios para S3 y AWS	30 de abril de 2015
Nueva versión del SDK	Publicación de la versión 3 del AWS SDK for PHP.	26 de mayo de 2015

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.