Panduan Pengembang untuk versi 3

AWS SDK for PHP

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

AWS SDK for PHP: Panduan Pengembang untuk versi 3

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS SDK for PHP? 1
Memulai dengan SDK 1
Sumber daya tambahan 1
Dokumentasi API 2
Pemeliharaan dan dukungan untuk versi utama SDK 2
Memulai 3
Prasyarat 3
Persyaratan 3
Rekomendasi 3
Uji kompatibilitas 5
Menginstal 5
Instal AWS SDK for PHP sebagai dependensi melalui Composer 5
Instalasi dengan menggunakan phar yang dikemas 6
Instalasi dengan menggunakan file ZIP 7
Autentikasi dengan AWS 7
Menyiapkan otentikasi dengan IAM Identity Center untuk pengembangan lokal 8
Memulai sesi portal AWS akses 9
Pelajari lebih lanjut tentang otentikasi 9
Membuat aplikasi sederhana 10
Prasyarat 10
Menyertakan SDK dalam kode Anda 10
Tulis kode 11
Menjalankan program 11
Langkah selanjutnya 12
Gunakan AWS Cloud9 dengan SDK 12
Langkah 1: Siapkan Akun AWS untuk Anda gunakan AWS Cloud9 12
Langkah 2: Siapkan lingkungan AWS Cloud9 pengembangan Anda 13
Langkah 3: Mengatur AWS SDK for PHP 13
Langkah 4: Unduh kode contoh 14
Langkah 5: Jalankan kode contoh 15
Mengkonfigurasi klien layanan 17
Konfigurasi klien secara eksternal 17
Rantai penyedia konfigurasi untuk konfigurasi klien 18
Buat klien layanan yang dikonfigurasi menggunakan pengaturan eksternal 18

AWS SDK for PHP Variabel lingkungan versi 3 19
Konfigurasi klien dalam kode 20
Konfigurasi dasar dalam kode 20
Menggunakan Sdk Kelas 21
Opsi konstruktor 22
Wilayah AWS 50
Rantai resolusi wilayah 51
Praktik terbaik 52
Penyedia kredensi 52
Apa yang dimaksud dengan penyedia kredensi di AWS SDK for PHP versi 3? 53
Memahami rantai penyedia kredensi default di AWS SDK for PHP Versi 3 54
Penyedia kredensi bawaan di AWS SDK for PHP Versi 3 55
Merantai penyedia kredensi di SDK for PHP 66
Membuat penyedia kredensi khusus untuk digunakan dengan SDK for PHP 67
Memoisasi kredensi di SDK for PHP 68
Asumsikan peran IAM 69
Menggunakan kredensial sementara dari AWS STS SDK for PHP 76
Buat klien anonim di SDK for PHP 78
Gunakan ekstensi AWS CRT 79
Apakah saya memerlukan ekstensi AWS CRT? 79
Bagaimana cara menginstal ekstensi AWS CRT? 79
Menggunakan SDK 80
Membuat Layanan AWS permintaan 80
Ikhtisar alur kerja permintaan SDK 80
Membuat klien layanan dasar 81
Membuat permintaan 82
Bekerja dengan objek hasil 83
Objek perintah 84
Pemrograman asinkron 92
Janji 94
Menangani kesalahan 101
Penanganan Kesalahan Sinkron 101
Penanganan kesalahan asinkron 102
Handler dan middleware 103
Penangan 103
Middleware 105

Membuat handler kustom 113
Pengaliran 114
Dekorator aliran 115
Paginasi 119
Objek paginator 120
Menghitung data dari hasil 120
pagination asinkron 121
Pelayan 122
Konfigurasi pelayan 123
Menunggu secara asinkron 124
JMESPath ekspresi 125
Mengekstrak data dari hasil 126
Mengekstrak data dari paginator 130
Memanggil Layanan AWS 132
Gunakan fitur dan opsi 132
Amazon DynamoDB 132
Amazon S3 139
Contoh kode dengan panduan 223
Kredensial 223
CloudFront Contoh Amazon 224
Amazon CloudSearch 253
CloudWatch Contoh Amazon 255
EC2 Contoh Amazon 280
OpenSearch Layanan Amazon 293
AWS Identity and Access Management contoh 295
AWS Key Management Service 320
Contoh Kinesis 342
AWS Elemental MediaConvert 358
Contoh Amazon S3 365
AWS Secrets Manager 400
Contoh Amazon SES 409
Contoh Amazon SNS 441
Contoh Amazon SQS 461
Amazon EventBridge 473
Contoh kode 475
API Gateway 476

Tindakan 476
Skenario 482
Aurora 482
Skenario 482
Auto Scaling 483
Memulai 484
Hal-hal mendasar 485
Tindakan 476
Amazon Bedrock 499
Tindakan 476
Runtime Amazon Bedrock 500
Skenario 482
Amazon Nova 503
Generator Gambar Amazon Titan 505
Antropik Claude 506
Difusi Stabil 507
Amazon DocumentDB 509
Contoh nirserver 509
DynamoDB 510
Hal-hal mendasar 485
Tindakan 476
Skenario 482
Contoh nirserver 509
Amazon EC2 543
Tindakan 476
AWS Glue 548
Hal-hal mendasar 485
Tindakan 476
IAM 568
Hal-hal mendasar 485
Tindakan 476
Kinesis 585
Contoh nirserver 509
AWS KMS 589
Memulai 484
Hal-hal mendasar 485

Tindakan 476
Lambda 625
Hal-hal mendasar 485
Tindakan 476
Skenario 482
Contoh nirserver 509
Amazon MSK 655
Contoh nirserver 509
Amazon RDS 657
Tindakan 476
Skenario 482
Contoh nirserver 509
Layanan Data Amazon RDS 665
Skenario 482
Amazon Rekognition 667
Skenario 482
Amazon S3 668
Memulai 484
Hal-hal mendasar 485
Tindakan 476
Skenario 482
Contoh nirserver 509
Ember Direktori S3 691
Hal-hal mendasar 485
Amazon SES 707
Skenario 482
Amazon SNS 708
Tindakan 476
Skenario 482
Contoh nirserver 509
Amazon SQS 729
Contoh nirserver 509
Migrasi dari Versi 2 733
Pengantar 733
Apa yang baru di versi 3? 733
Lapisan HTTP terpisah 733

Permintaan asinkron ... 92
Apa yang berbeda dari versi 2? ... 734
Dependensi proyek diperbarui ... 734
Opsi wilayah dan versi sekarang diperlukan ... 735
Instantiasi klien menggunakan konstruktor ... 735
Konfigurasi klien telah berubah ... 736
Beberapa hasil API telah berubah ... 737
Kelas Enum telah dihapus ... 740
Kelas pengecualian berbutir halus telah dihapus ... 740
Kelas fasad statis telah dihapus ... 741
Paginator menggantikan iterator ... 741
Banyak abstraksi tingkat tinggi telah berubah ... 742
Membandingkan sampel kode dari kedua versi SDK ... 743
Contoh: Operasi Amazon S3 `ListObjects` ... 743
Contoh: Membuat instantiasi klien dengan konfigurasi global ... 744
Keamanan ... 747
Perlindungan data ... 747
Identity and Access Management ... 748
Audiens ... 749
Mengautentikasi dengan identitas ... 750
Mengelola akses menggunakan kebijakan ... 751
Bagaimana Layanan AWS bekerja dengan IAM ... 753
Memecahkan masalah AWS identitas dan akses ... 753
Validasi Kepatuhan ... 755
Ketahanan ... 756
Keamanan Infrastruktur ... 756
Migrasi klien enkripsi Amazon S3 (V1 ke V2) ... 757
Ikhtisar migrasi ... 757
Perbarui klien yang ada untuk membaca format baru ... 757
Migrasikan enkripsi dan dekripsi klien ke V2 ... 758
Contoh migrasi ... 759
Migrasi klien enkripsi Amazon S3 (V2 ke V3) ... 762
Ikhtisar migrasi ... 762
Memahami konsep V3 ... 763
Perbarui klien yang ada untuk membaca format baru ... 765
Migrasikan klien Enkripsi dan Dekripsi ke V3 ... 766

Contoh tambahan 771
Pertanyaan yang Sering Diajukan 775
Metode apa yang tersedia pada klien? 775
Apa yang harus saya lakukan tentang kesalahan sertifikat cURL SSL? 775
Versi API apa yang tersedia untuk klien? 775
Versi Region apa yang tersedia untuk klien? 776
Mengapa saya tidak dapat mengunggah atau mengunduh file yang lebih besar dari 2 GB? 776
Bagaimana saya bisa melihat data apa yang dikirim melalui kabel? 776
Bagaimana saya bisa mengatur header arbitrer berdasarkan permintaan? 777
Bagaimana saya bisa menandatangani permintaan sewenang-wenang? 777
Bagaimana saya bisa memodifikasi perintah sebelum mengirimnya? 777
Apa itu CredentialsException? 777
Apakah AWS SDK for PHP pekerjaan pada HHVM? 778
Bagaimana cara menonaktifkan SSL? 778
Apa yang harus saya lakukan tentang “Parse error”? 779
Mengapa klien Amazon S3 mendekompresi file gzip? 779
Bagaimana cara menonaktifkan penandatanganan tubuh di Amazon S3? 779
Bagaimana skema coba lagi ditangani di? AWS SDK for PHP 780
Bagaimana cara menangani pengecualian dengan kode kesalahan? 780
Glosarium 782
Riwayat dokumen 785
..... dccxc

Apa itu AWS SDK for PHP versi 3?

AWS SDK for PHP Versi 3 memungkinkan pengembang PHP untuk menggunakan Amazon Web Services dalam kode PHP mereka, dan membangun aplikasi dan perangkat lunak yang kuat menggunakan layanan seperti Amazon S3, Amazon DynamoDB, dan Amazon Glacier. Anda dapat memulai dalam hitungan menit dengan menginstal SDK melalui Composer — dengan meminta `aws/aws-sdk-php` paket — atau dengan mengunduh file mandiri `aws.zip`atau file. `aws.phar`

Tidak semua layanan segera tersedia di SDK. Untuk mengetahui layanan mana yang saat ini didukung oleh AWS SDK for PHP, lihat Nama Layanan dan Versi API.

> **Note**
>
> Jika Anda memigrasikan kode Anda dari menggunakan Versi 2 SDK ke Versi 3, pastikan untuk membaca Upgrade dari Versi 2 dari versi. AWS SDK for PHP

Memulai dengan SDK

Jika Anda siap untuk langsung menggunakan SDK, ikuti bab-babnya. Memulai dengan AWS SDK for PHP Versi 3 Ini memandu Anda melalui otentikasi dengan AWS, menyiapkan lingkungan pengembangan Anda, dan membuat aplikasi dasar pertama Anda menggunakan Amazon S3.

Sumber daya tambahan

- FAQ
- Glosarium
- AWS SDKs dan Panduan Referensi Alat: Berisi pengaturan, fitur, dan konsep dasar lainnya yang umum di antara AWS SDKs.
- Dokumentasi Guzzle
- Contoh kode yang menggunakan AWS SDK for PHP tersedia di awsdocs/ aws-doc-sdk-examples repo.
- Komunitas PHP SDK di Gitter.
- AWS re:Post.

GitHub:

- Kode sumber untuk AWS SDK for PHP tersedia di aws/ aws-sdk-php repo.
- Berkontribusi pada SDK
- Laporkan bug atau minta fitur

Dokumentasi API

Temukan dokumentasi API untuk SDK https://docs.aws.amazon.com/sdk-for-php/terbaru/referensi/.

Pemeliharaan dan dukungan untuk versi utama SDK

Untuk informasi tentang pemeliharaan dan dukungan untuk versi utama SDK dan dependensi yang mendasarinya, lihat berikut ini di Panduan Referensi Alat AWS SDKs dan Alat berikut:

- AWS SDKs dan kebijakan pemeliharaan alat
- AWS SDKs dan matriks dukungan versi Tools

Memulai dengan AWS SDK for PHP Versi 3

Bab ini didedikasikan untuk membuat Anda bangun dan menjalankan dengan AWS SDK for PHP Versi 3.

Topik

- Persyaratan dan rekomendasi untuk AWS SDK for PHP Versi 3
- Menginstal AWS SDK for PHP Versi 3
- Mengautentikasi dengan AWS menggunakan AWS SDK for PHP Versi 3
- Membuat aplikasi sederhana menggunakan AWS SDK for PHP Versi 3
- Gunakan AWS Cloud9 dengan AWS SDK for PHP Versi 3

Persyaratan dan rekomendasi untuk AWS SDK for PHP Versi 3

Untuk hasil terbaik dengan AWS SDK for PHP, pastikan lingkungan Anda mendukung persyaratan dan rekomendasi berikut.

Persyaratan

Untuk menggunakan AWS SDK for PHP, Anda harus menggunakan PHP versi 8.1 atau yang lebih baru dengan ekstensi SimpleXMLPHP diaktifkan. Jika Anda perlu menandatangani Amazon pribadi CloudFront URLs, Anda juga memerlukan ekstensi PHP OpenSSL.

Rekomendasi

Selain persyaratan minimum, kami sarankan Anda juga menginstal, menghapus, dan menggunakan yang berikut ini.

Instal cURL 7.16.2 atau yang lebih baru	Gunakan cURL versi terbaru yang dikompilasi dengan OpenSSL/NSS dan zlib. Jika cURL tidak diinstal pada sistem Anda dan Anda tidak mengkonfigurasi http_handler kustom untuk klien Anda, SDK menggunakan pembungkus aliran PHP.

Gunakan OPCache	Gunakan OPcache ekstensi untuk meningkatkan kinerja PHP dengan menyimpan bytecode skrip yang telah dikompilasi dalam memori bersama. Ini menghilangkan kebutuhan PHP untuk memuat dan mengurai skrip pada setiap permintaan. Ekstensi ini biasanya diaktifkan secara default. Saat menjalankan Amazon Linux, Anda perlu menginstal paket yum php56-opcache atau php55-opcache untuk menggunakan ekstensi. OPCache
Copot pemasangan Xdebug di lingkungan produksi	Xdebug dapat membantu mengidentifikasi kemacetan kinerja. Namun, jika kinerja sangat penting untuk aplikasi Anda, jangan instal ekstensi Xdebug di lingkungan produksi Anda. Memuat ekstensi sangat memperlambat kinerja SDK.
Menggunakan autoloader classmap Composer	Autoloader memuat kelas seperti yang dibutuhkan oleh skrip PHP. Composer menghasilkan autoloader yang dapat memuat otomatis skrip PHP aplikasi Anda dan semua skrip PHP lainnya yang diperlukan oleh aplikasi Anda, termasuk file. AWS SDK for PHP Untuk lingkungan produksi, kami menyarankan Anda menggunakan autoloader classmap untuk meningkatkan performa autoloader. Anda dapat membuat autoloader classmap dengan meneruskan `==optimize-autoloader` opsi `-o` or ke perintah install Composer.

Uji kompatibilitas

Jalankan `compatibility-test.php`file yang terletak di basis kode SDK untuk memverifikasi sistem Anda dapat menjalankan SDK. Selain memenuhi persyaratan sistem minimum SDK, uji kompatibilitas memeriksa pengaturan opsional dan membuat rekomendasi yang dapat membantu meningkatkan kinerja. Hasil uji kompatibilitas menghasilkan baik ke baris perintah atau browser web. Saat meninjau hasil pengujian di browser, pemeriksaan yang berhasil muncul dalam warna hijau, peringatan berwarna ungu, dan kegagalan berwarna merah. Saat menjalankan dari baris perintah, hasil pemeriksaan muncul pada baris terpisah.

Saat melaporkan masalah dengan SDK, membagikan output uji kompatibilitas membantu mengidentifikasi penyebab yang mendasarinya.

Menginstal AWS SDK for PHP Versi 3

Anda dapat menginstal AWS SDK for PHP versi 3:

- Sebagai ketergantungan melalui Komposer
- Sebagai phar SDK yang dikemas sebelumnya
- Sebagai file ZIP dari SDK

Sebelum Anda menginstal AWS SDK for PHP Versi 3 pastikan lingkungan Anda menggunakan PHP versi 8.1 atau yang lebih baru. Pelajari lebih lanjut tentang persyaratan dan rekomendasi lingkungan.

> **Note**
>
> Menginstal SDK melalui metode.phar dan .zip memerlukan ekstensi PHP Multibyte String untuk diinstal dan diaktifkan secara terpisah.

Instal AWS SDK for PHP sebagai dependensi melalui Composer

Komposer adalah cara yang disarankan untuk menginstal. AWS SDK for PHP Composer adalah alat untuk PHP yang mengelola dan menginstal dependensi proyek Anda.

Untuk informasi selengkapnya tentang cara menginstal Composer, mengonfigurasi pemuatan otomatis, dan mengikuti praktik terbaik lainnya untuk menentukan dependensi, lihat getcomposer.org.

Instal Composer

Jika Composer belum ada di proyek Anda, unduh dan instal Komposer di halaman Download Composer.

- Untuk Windows, ikuti instruksi Penginstal Windows.
- Untuk Linux, ikuti instruksi instalasi baris perintah.

Tambahkan AWS SDK for PHP sebagai ketergantungan melalui Komposer

Jika Composer sudah diinstal secara global pada sistem Anda, jalankan yang berikut ini di direktori dasar proyek Anda untuk menginstal AWS SDK for PHP sebagai dependensi:

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

Jika tidak, ketik perintah Composer ini untuk menginstal versi terbaru dari AWS SDK for PHP sebagai dependensi.

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

Tambahkan autoloader ke skrip php Anda

Menginstal Composer membuat beberapa folder dan file di lingkungan Anda. File utama yang akan Anda gunakan adalah `autoload.php`, yang ada di folder `vendor` di lingkungan Anda.

Untuk memanfaatkan skrip Anda, sertakan autoloader dalam skrip Anda, sebagai berikut. AWS SDK for PHP

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

Instalasi dengan menggunakan phar yang dikemas

Setiap rilis AWS SDK for PHP menyertakan phar prepackaged (arsip PHP) yang berisi semua kelas dan dependensi yang Anda butuhkan untuk menjalankan SDK. Selain itu, phar secara otomatis mendaftarkan autoloader kelas untuk AWS SDK for PHP dan semua dependensinya.

Anda dapat mengunduh phar yang dikemas dan memasukkannya ke dalam skrip Anda.

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

> **Note**
>
> Menggunakan PHP dengan patch Suhosin tidak dianjurkan, tetapi umum pada distribusi Ubuntu dan Debian. Dalam hal ini, Anda mungkin perlu mengaktifkan penggunaan phars di suhosin.ini. Jika Anda tidak melakukan ini, menyertakan file phar dalam kode Anda akan menyebabkan kegagalan diam. Untuk memodifikasi suhosin.ini, tambahkan baris berikut.
>
> ```
> suhosin.executor.include.whitelist = phar
> ```

Instalasi dengan menggunakan file ZIP

AWS SDK for PHP Termasuk file ZIP yang berisi semua kelas dan dependensi yang Anda butuhkan untuk menjalankan SDK. Selain itu, file ZIP menyertakan autoloader kelas untuk AWS SDK for PHP dan dependensinya.

Untuk menginstal SDK, unduh file.zip, lalu ekstrak ke proyek Anda di lokasi yang Anda pilih. Kemudian sertakan autoloader dalam skrip Anda, sebagai berikut.

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

Mengautentikasi dengan AWS menggunakan AWS SDK for PHP Versi 3

Anda harus menetapkan bagaimana kode Anda mengautentikasi AWS saat mengembangkan dengan Layanan AWS. Anda dapat mengonfigurasi akses terprogram ke AWS sumber daya dengan cara yang berbeda tergantung pada lingkungan dan AWS akses yang tersedia untuk Anda.

Untuk memilih metode otentikasi dan mengonfigurasinya untuk SDK, lihat Autentikasi dan akses di Panduan Referensi Alat AWS SDKs dan Alat.

Menyiapkan otentikasi dengan IAM Identity Center untuk pengembangan lokal

Kami menyarankan agar pengguna baru yang berkembang secara lokal dan tidak diberi metode otentikasi oleh majikan mereka harus disiapkan. AWS IAM Identity Center Metode ini termasuk menginstal AWS CLI untuk kemudahan konfigurasi dan untuk masuk secara teratur ke portal AWS akses. Jika Anda memilih metode ini, lingkungan Anda harus berisi elemen-elemen berikut setelah Anda menyelesaikan prosedur untuk autentikasi IAM Identity Center di AWS SDKs dan Panduan Referensi Alat:

- Itu AWS CLI, yang Anda gunakan untuk memulai sesi portal AWS akses sebelum Anda menjalankan aplikasi Anda.
- AWSconfigFile bersama yang memiliki `[default]` profil dengan serangkaian nilai konfigurasi yang dapat direferensikan oleh SDK. Untuk menemukan lokasi file ini, lihat Lokasi file bersama di Panduan Referensi Alat AWS SDKs dan.
- `config`File bersama berisi `region`pengaturan. Ini menetapkan default Wilayah AWS yang digunakan SDK untuk permintaan. Wilayah ini digunakan untuk permintaan layanan SDK yang tidak dikonfigurasi secara eksplisit dengan properti. `region`
- SDK menggunakan konfigurasi penyedia token SSO profil untuk memperoleh kredensil sebelum mengirim permintaan ke. AWS`sso_role_name`Nilai, yang merupakan peran IAM yang terhubung ke set izin Pusat Identitas IAM, memungkinkan akses ke yang Layanan AWS digunakan dalam aplikasi Anda.

  `config`File contoh berikut menunjukkan profil default yang diatur dengan konfigurasi penyedia token SSO. `sso_session`Pengaturan profil mengacu pada `sso-session`bagian bernama. `sso-session`Bagian ini berisi pengaturan untuk memulai sesi portal AWS akses.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

Tidak AWS SDK for PHP memerlukan paket tambahan (seperti `SSO` dan`SSOOIDC`) untuk ditambahkan ke aplikasi Anda untuk menggunakan autentikasi IAM Identity Center.

Memulai sesi portal AWS akses

Sebelum menjalankan aplikasi yang mengakses Layanan AWS, Anda memerlukan sesi portal AWS akses aktif agar SDK menggunakan autentikasi IAM Identity Center untuk menyelesaikan kredensialnya. Bergantung pada panjang sesi yang dikonfigurasi, akses Anda pada akhirnya akan kedaluwarsa dan SDK akan mengalami kesalahan otentikasi. Untuk masuk ke portal AWS akses, jalankan perintah berikut di AWS CLI.

```
aws sso login
```

Jika Anda mengikuti panduan dan memiliki pengaturan profil default, Anda tidak perlu memanggil perintah dengan `--profile` opsi. Jika konfigurasi penyedia token SSO Anda menggunakan profil bernama, perintahnya adalah`aws sso login --profile named-profile`.

Untuk menguji secara opsional apakah Anda sudah memiliki sesi aktif, jalankan AWS CLI perintah berikut.

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

Jika sesi Anda aktif, respons terhadap perintah ini melaporkan akun Pusat Identitas IAM dan set izin yang dikonfigurasi dalam `config` file bersama.

> **Note**
>
> Jika Anda sudah memiliki sesi portal AWS akses aktif dan menjalankannya`aws sso login`, Anda tidak akan diminta untuk memberikan kredensil.
> Proses masuk mungkin meminta Anda untuk mengizinkan AWS CLI akses ke data Anda. Karena AWS CLI dibangun di atas SDK untuk Python, pesan izin mungkin berisi variasi nama. `botocore`

Pelajari lebih lanjut tentang otentikasi

- Untuk detail selengkapnya tentang menggunakan Pusat Identitas IAM untuk autentikasi, lihat Memahami autentikasi Pusat Identitas IAM di Panduan Referensi Alat dan AWS SDKs

- Untuk mempelajari lebih lanjut tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.
- Untuk membuat AWS kredensil jangka pendek, lihat Kredenal Keamanan Sementara di Panduan Pengguna IAM.
- Untuk mempelajari tentang penyedia kredensi lain yang AWS SDK for PHP dapat digunakan, lihat Penyedia kredensi terstandarisasi di Panduan Referensi Alat AWS SDKs dan Alat.

Membuat aplikasi sederhana menggunakan AWS SDK for PHP Versi 3

Sapa Amazon S3 menggunakan file. AWS SDK for PHP Contoh berikut menampilkan daftar bucket Amazon S3 Anda.

Prasyarat

- Unduh dan instal SDK
- Sebelum Anda menggunakan AWS SDK for PHP, Anda harus mengatur otentikasi dengan AWS. Untuk informasi tentang menyiapkan autentikasi, lihat Mengautentikasi dengan AWS menggunakan AWS SDK for PHP Versi 3

Menyertakan SDK dalam kode Anda

Apa pun teknik yang Anda gunakan untuk menginstal SDK, Anda dapat menyertakan SDK dalam kode Anda hanya dengan satu `require` pernyataan. Lihat tabel berikut untuk kode PHP yang paling sesuai dengan teknik instalasi Anda. Ganti setiap instance `/path/to/` dengan jalur aktual di sistem Anda.

Teknik Instalasi	Memerlukan Pernyataan
Menggunakan Komposer	`require '/path/to/vendor/autoload.php';`
Menggunakan phar	`require '/path/to/aws.phar';`
Menggunakan ZIP	`require '/path/to/aws-autoloader.php';`

Dalam topik ini, kita mengasumsikan metode instalasi Composer. Jika Anda menggunakan metode instalasi yang berbeda, Anda dapat merujuk kembali ke bagian ini untuk menemukan `require` kode yang benar untuk digunakan.

Tulis kode

Pastikan Anda dapat mengautentikasi.

Salin dan tempel kode berikut ke file sumber baru. Simpan dan beri nama file`hello-s3.php`.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

//Create a S3Client
// snippet-start:[s3.php.list_buckets.main]
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Listing all S3 Bucket
$buckets = $s3Client->listBuckets();
foreach ($buckets['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}
```

Menjalankan program

Buka command prompt untuk menjalankan program PHP Anda. Sintaks perintah khas untuk menjalankan program PHP adalah:

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

Kode contoh ini tidak menggunakan argumen. Untuk menjalankan kode ini, masukkan yang berikut ini ke command prompt:

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

Langkah selanjutnya

Untuk menguji banyak operasi Amazon S3 lainnya, lihat Repositori Contoh AWS Kode di. GitHub

Gunakan AWS Cloud9 dengan AWS SDK for PHP Versi 3

> **Note**
>
> AWS Cloud9 tidak lagi tersedia untuk pelanggan baru. Pelanggan yang sudah ada AWS Cloud9 dapat terus menggunakan layanan seperti biasa. Pelajari selengkapnya.

AWS Cloud9 adalah lingkungan pengembangan terintegrasi berbasis web (IDE) yang berisi kumpulan alat yang Anda gunakan untuk membuat kode, membangun, menjalankan, menguji, men-debug, dan merilis perangkat lunak di cloud. Anda dapat menggunakan AWS Cloud9 dengan AWS SDK for PHP untuk menulis dan menjalankan kode PHP Anda dengan menggunakan browser. AWS Cloud9 termasuk alat-alat seperti editor kode dan terminal. Karena AWS Cloud9 IDE berbasis cloud, Anda dapat mengerjakan proyek Anda dari kantor, rumah, atau di mana saja dengan menggunakan mesin yang terhubung ke internet. Untuk informasi umum tentang AWS Cloud9, lihat Panduan AWS Cloud9 Pengguna.

Ikuti petunjuk ini untuk mengatur AWS Cloud9 dengan AWS SDK for PHP:

- Langkah 1: Siapkan Akun AWS untuk Anda gunakan AWS Cloud9
- Langkah 2: Siapkan lingkungan AWS Cloud9 pengembangan Anda
- Langkah 3: Mengatur AWS SDK for PHP
- Langkah 4: Unduh kode contoh
- Langkah 5: Jalankan kode contoh

Langkah 1: Siapkan Akun AWS untuk Anda gunakan AWS Cloud9

Untuk menggunakan AWS Cloud9, masuk ke AWS Cloud9 konsol dari file Konsol Manajemen AWS.

> **Note**
>
> Jika Anda menggunakan AWS IAM Identity Center untuk mengautentikasi, Anda mungkin perlu menambahkan izin yang diperlukan `iam:ListInstanceProfilesForRole` ke kebijakan yang dilampirkan pengguna di konsol IAM.

Untuk menyiapkan entitas IAM di AWS akun Anda agar dapat mengakses AWS Cloud9 dan masuk ke AWS Cloud9 konsol, lihat Pengaturan Tim AWS Cloud9 di Panduan AWS Cloud9 Pengguna.

Langkah 2: Siapkan lingkungan AWS Cloud9 pengembangan Anda

Setelah Anda masuk ke AWS Cloud9 konsol, gunakan konsol untuk membuat lingkungan AWS Cloud9 pengembangan. Setelah Anda membuat lingkungan, AWS Cloud9 buka IDE untuk lingkungan itu.

Untuk detailnya, lihat Membuat Lingkungan AWS Cloud9 di Panduan AWS Cloud9 Pengguna.

> **Note**
>
> Saat Anda membuat lingkungan di konsol untuk pertama kalinya, kami sarankan Anda memilih opsi untuk Membuat instance baru untuk environment (EC2). Opsi ini memberi tahu AWS Cloud9 untuk membuat lingkungan, meluncurkan EC2 instance Amazon, dan kemudian menghubungkan instance baru ke lingkungan baru. Ini adalah cara tercepat untuk mulai menggunakan AWS Cloud9.

Jika terminal belum terbuka di IDE, buka. Pada bilah menu di IDE, pilih Jendela, Terminal Baru. Anda dapat menggunakan jendela terminal untuk menginstal alat dan membangun aplikasi Anda.

Langkah 3: Mengatur AWS SDK for PHP

Setelah AWS Cloud9 membuka IDE untuk lingkungan pengembangan Anda, gunakan jendela terminal untuk mengatur AWS SDK for PHP di lingkungan Anda.

Komposer adalah cara yang disarankan untuk menginstal. AWS SDK for PHP Composer adalah alat untuk PHP yang mengelola dan menginstal dependensi proyek Anda.

Untuk informasi selengkapnya tentang cara menginstal Composer, mengonfigurasi pemuatan otomatis, dan mengikuti praktik terbaik lainnya untuk menentukan dependensi, lihat getcomposer.org.

Instal Composer

Jika Komposer belum ada di proyek Anda, unduh dan instal Komposer di halaman Unduh Komposer.

- Untuk Windows, ikuti instruksi Penginstal Windows.
- Untuk Linux, ikuti instruksi instalasi baris perintah.

Tambahkan AWS SDK for PHP sebagai ketergantungan melalui Komposer

Jika Composer sudah diinstal secara global pada sistem Anda, jalankan yang berikut ini di direktori dasar proyek Anda untuk menginstal AWS SDK for PHP sebagai dependensi:

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

Jika tidak, ketik perintah Composer ini untuk menginstal versi terbaru dari AWS SDK for PHP sebagai dependensi.

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

Tambahkan autoloader ke skrip php Anda

Menginstal Composer membuat beberapa folder dan file di lingkungan Anda. File utama yang akan Anda gunakan adalah `autoload.php`, yang ada di folder `vendor` di lingkungan Anda.

Untuk memanfaatkan skrip Anda, sertakan autoloader dalam skrip Anda, sebagai berikut. AWS SDK for PHP

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

Langkah 4: Unduh kode contoh

Gunakan jendela terminal untuk mengunduh kode contoh untuk AWS SDK for PHP ke dalam lingkungan AWS Cloud9 pengembangan.

Untuk mengunduh salinan semua contoh kode yang digunakan dalam dokumentasi AWS SDK resmi ke direktori root lingkungan Anda, jalankan perintah berikut:

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

Contoh kode untuk AWS SDK for PHP terletak di `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` direktori, di mana `ENVIRONMENT_NAME` adalah nama lingkungan pengembangan Anda.

Untuk mengikuti menggunakan contoh Amazon S3, sebaiknya mulai dengan contoh kode. `ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php` Contoh ini akan mencantumkan bucket Amazon S3 Anda. Gunakan jendela terminal untuk menavigasi ke `s3` direktori dan daftar file.

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

Untuk membuka file di AWS Cloud9, Anda dapat mengklik `ListBuckets.php` langsung di jendela terminal.

Untuk dukungan lebih lanjut dalam memahami contoh kode, lihat Contoh AWS SDK for PHP Kode.

Langkah 5: Jalankan kode contoh

Untuk menjalankan kode di lingkungan AWS Cloud9 pengembangan Anda, pilih tombol Run di bilah menu atas. AWS Cloud9 secara otomatis mendeteksi ekstensi `.php` file dan menggunakan pelari PHP (server web bawaan) untuk menjalankan kode. Namun, untuk contoh ini kita sebenarnya menginginkan opsi PHP (**`cli`**). Untuk informasi selengkapnya tentang menjalankan kode AWS Cloud9, lihat Menjalankan Kode Anda di Panduan AWS Cloud9 Pengguna.

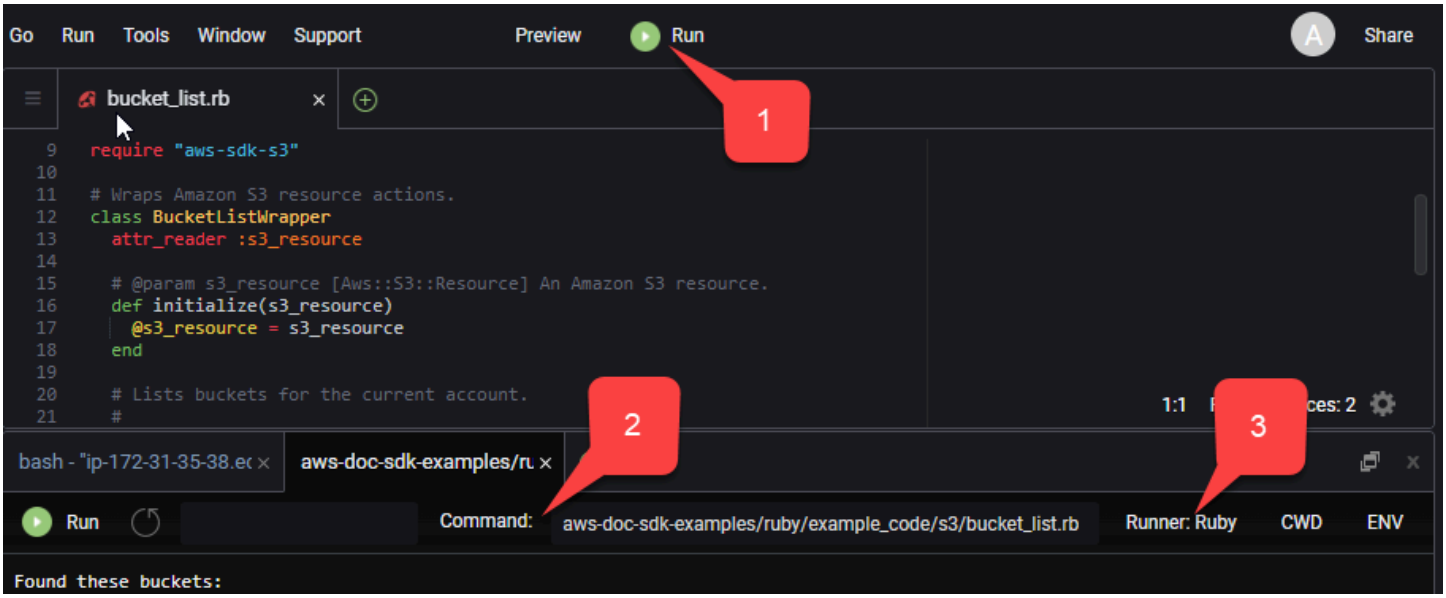
Pada tangkapan layar berikut, perhatikan area dasar ini:

- 1: Lari. Tombol Run terletak di bilah menu atas. Ini membuka tab baru untuk hasil Anda.

  > **Note**
  >
  > Anda juga dapat membuat konfigurasi run baru secara manual. Pada bilah menu, pilih Jalankan, Jalankan Konfigurasi, Jalankan Konfigurasi Baru.

- 2: Perintah. AWS Cloud9 mengisi kotak teks Command dengan path dan nama file ke file yang Anda jalankan. Jika kode Anda mengharapkan parameter baris perintah apa pun yang akan diteruskan, ini dapat ditambahkan ke baris perintah dengan cara yang sama seperti yang Anda lakukan saat menjalankan kode melalui jendela terminal.

- 3: Pelari. AWS Cloud9 mendeteksi bahwa ekstensi file Anda `.php` dan memilih PHP (built-in web server) Runner untuk menjalankan kode Anda. Pilih PHP (**`cli`**) untuk menjalankan contoh ini sebagai gantinya.



Setiap output yang dihasilkan dari kode yang sedang berjalan ditampilkan di tab.

Mengkonfigurasi klien layanan di AWS SDK for PHP Versi 3

Untuk mengakses secara terprogram Layanan AWS, AWS SDK for PHP Versi 3 menggunakan objek klien untuk masing-masing. Layanan AWS Misalnya, jika aplikasi Anda perlu mengakses Amazon EC2, Anda membuat objek EC2 klien Amazon—instance kelas Ec2Client—untuk berinteraksi dengan layanan tersebut. Anda kemudian menggunakan klien layanan untuk membuat permintaan untuk itu Layanan AWS.

Ada banyak cara untuk mengonfigurasi perilaku SDK, tetapi pada akhirnya semuanya berkaitan dengan perilaku klien layanan. Konfigurasi apa pun tidak berpengaruh sampai kode Anda membuat klien layanan yang menggunakan konfigurasi.

Contoh konfigurasi yang Anda berikan adalah:

- Bagaimana kode Anda mengautentikasi AWS ketika Anda memanggil layanan
- Wilayah AWS Anda ingin klien layanan untuk menggunakan
- Coba lagi dan pengaturan batas waktu untuk panggilan layanan
- Konfigurasi proxy HTTP

Lihat Panduan Referensi AWS SDKs and Tools untuk pengaturan, fitur, dan konsep dasar lainnya yang umum untuk banyak. AWS SDKs

Mengkonfigurasi klien layanan untuk AWS SDK for PHP Versi 3 secara eksternal

Banyak pengaturan konfigurasi dapat ditangani di luar kode Anda. Sebagian besar pengaturan konfigurasi dapat diatur sebagai variabel lingkungan atau dalam AWS `config` file bersama yang terpisah. `config`File AWS bersama dapat mempertahankan set pengaturan terpisah, yang disebut profil, untuk menyediakan konfigurasi yang berbeda untuk lingkungan atau pengujian yang berbeda. Untuk diskusi lengkap tentang AWS berbagi `config` dan `credentials` file, lihat Panduan Referensi Alat AWS SDKs dan Alat.

Sebagian besar variabel lingkungan dan pengaturan `config` file bersama distandarisasi dan dibagikan di seluruh AWS SDKs dan alat untuk mendukung fungsionalitas yang konsisten di berbagai bahasa dan aplikasi pemrograman.

Untuk melihat semua pengaturan yang dapat diselesaikan SDK dari variabel lingkungan atau file konfigurasi, lihat referensi Pengaturan di Panduan Referensi Alat AWS SDKs dan Alat.

Rantai penyedia konfigurasi untuk konfigurasi klien

SDK memeriksa beberapa tempat (atau sumber) untuk menemukan nilai konfigurasi.

1. Pengaturan eksplisit apa pun yang disetel dalam kode atau pada klien layanan itu sendiri lebih diutamakan daripada yang lain.
2. Variabel-variabel lingkungan
   - Untuk detail tentang pengaturan variabel lingkungan, lihat variabel lingkungan di Panduan Referensi Alat AWS SDKs dan Alat.
   - Perhatikan bahwa Anda dapat mengonfigurasi variabel lingkungan untuk shell pada tingkat cakupan yang berbeda: seluruh sistem, seluruh pengguna, dan untuk sesi terminal tertentu.
3. Berbagi `config` dan `credentials` file
   - Untuk detail tentang menyiapkan file-file ini, lihat Dibagikan `config` dan `credentials` file di Panduan Referensi Alat AWS SDKs dan Alat.
4. Setiap nilai default yang disediakan oleh kode sumber SDK itu sendiri digunakan terakhir.
   - Beberapa properti, seperti Region, tidak memiliki default. Anda harus menentukannya secara eksplisit dalam kode, dalam pengaturan lingkungan, atau dalam file bersama`config`. Jika SDK tidak dapat menyelesaikan konfigurasi yang diperlukan, permintaan API dapat gagal saat runtime.

Selain rantai konfigurasi umum ini, AWS SDK for PHP Versi 3 juga menggunakan rantai penyedia khusus termasuk rantai penyedia kredensil dan rantai Wilayah AWS resolusi. Rantai khusus ini menambahkan penyedia tambahan yang memperhitungkan lingkungan tempat SDK berjalan. Misalnya, dalam wadah atau EC2 instance.

Buat klien layanan yang dikonfigurasi menggunakan pengaturan eksternal

Anda perlu membuat klien layanan di aplikasi Anda untuk berbicara dengan Layanan AWS. Pelanggan layanan adalah koneksi penting Anda Layanan AWS, menangani semua detail komunikasi yang kompleks sehingga Anda tidak perlu khawatir tentang mereka. Mereka menangani tugas-tugas penting seperti keamanan, penanganan kesalahan, dan percobaan ulang secara otomatis, memungkinkan Anda fokus pada membangun aplikasi Anda daripada berurusan dengan komplikasi teknis.

Membangun klien layanan menggunakan konstruktor tanpa parameter

Jika semua pengaturan konfigurasi yang Anda butuhkan berasal dari sumber eksternal, Anda dapat membuat klien layanan dengan menggunakan konstruktor kosong:

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

use Aws\S3\S3Client;

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

Cuplikan kode sebelumnya membuat sebuah `S3Client` instance. Selama pembuatan, SDK melihat melalui rantai penyedia konfigurasi untuk pengaturan. Setelah SDK menemukan nilai setelan, SDK menggunakan nilainya.

Rantai Wilayah AWS penyedia default dan rantai penyedia kredensi default juga akan digunakan dalam proses pembuatan. Di suatu tempat dalam rantai, SDK harus menyelesaikan pengaturan yang akan digunakan dan menemukan yang Wilayah AWS memungkinkannya mengambil kredensil untuk permintaan penandatanganan. Jika SDK gagal menemukan nilai tersebut, pembuatan klien gagal.

AWS SDK for PHP Variabel lingkungan versi 3

Di luar pengaturan cross-sdk yang didukung oleh sebagian besar AWS SDKs, AWS SDK for PHP Versi 3 bekerja dengan variabel lingkungan berikut:

AWS_SDK_LOAD_NONDEFAULT_CONFIG

Saat disetel, variabel lingkungan ini menginstruksikan SDK untuk memuat kredensil dari AWS `config` file (`~/.aws/config`) selain `credentials` file. `~/.aws/credential`

AWS_SDK_UA_APP_ID

Menetapkan pengenal aplikasi kustom yang disertakan dalam header User-Agent untuk permintaan yang dibuat oleh SDK.

AWS_SUPPRESS_PHP_DEPRECATION_WARNING

Ketika disetel ke true, menekan peringatan penghentian PHP yang mungkin dihasilkan SDK.

Mengkonfigurasi klien layanan dalam kode untuk AWS SDK for PHP Versi 3

Sebagai alternatif untuk—atau selain— mengkonfigurasi klien layanan secara eksternal, Anda dapat mengonfigurasinya secara terprogram dalam kode.

Dengan mengonfigurasi klien layanan dalam kode, Anda mendapatkan kontrol halus dari banyak opsi yang tersedia untuk Anda. Sebagian besar konfigurasi yang dapat Anda atur secara eksternal juga tersedia untuk Anda atur dalam kode.

Konfigurasi dasar dalam kode

Anda dapat membuat dan mengkonfigurasi klien layanan dalam kode dengan meneruskan array pilihan asosiatif ke konstruktor klien. Dalam contoh berikut, array asosiatif hanya berisi opsi “wilayah” yang digunakan klien:

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

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

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

Informasi tentang parameter “versi” opsional tersedia di topik opsi konfigurasi.

Perhatikan bahwa kami tidak secara eksplisit memberikan kredensional kepada klien. Itu karena SDK menggunakan rantai penyedia kredensi default untuk mencari informasi kredensi.

Semua opsi konfigurasi klien umum dijelaskan secara rinci diOpsi konstruktor klien untuk AWS SDK for PHP Versi 3. Array opsi yang diberikan kepada klien dapat bervariasi berdasarkan klien mana yang Anda buat. Opsi konfigurasi klien kustom ini dijelaskan dalam dokumentasi API untuk setiap klien.

Menggunakan **Sdk** Kelas

`Aws\Sdk`Kelas bertindak sebagai pabrik klien dan digunakan untuk mengelola opsi konfigurasi bersama di beberapa klien. Banyak opsi yang dapat diberikan kepada konstruktor klien tertentu juga dapat diberikan ke `Aws\Sdk` kelas. Opsi ini kemudian diterapkan ke setiap konstruktor klien.

Impor

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

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

Kode Sampel

```
// The same options that can be provided to a specific client constructor can also be
 supplied to the Aws\Sdk class.
// Use the us-west-2 region and latest version of each client.
$sharedConfig = [
    'region' => 'us-west-2'
];
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk($sharedConfig);
// Create an Amazon S3 client using the shared configuration data.
$client = $sdk->createS3();
```

Opsi yang dibagikan di semua klien ditempatkan dalam pasangan nilai kunci tingkat root. Data konfigurasi khusus layanan dapat disediakan dalam array asosiatif dengan kunci yang sama dengan namespace layanan (misalnya, “S3", “”, dll.). DynamoDb

```
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2',
    'DynamoDb' => [
        'region' => 'eu-central-1'
    ]
]);

// Creating an Amazon DynamoDb client will use the "eu-central-1" AWS Region.
$client = $sdk->createDynamoDb();
```

Nilai konfigurasi khusus layanan adalah gabungan dari nilai spesifik layanan dan nilai tingkat root (yaitu, nilai khusus layanan digabungkan secara dangkal ke nilai tingkat root).

> **Note**
>
> Kami sangat menyarankan Anda menggunakan `Sdk` kelas untuk membuat klien jika Anda menggunakan beberapa instance klien dalam aplikasi Anda. `Sdk`Kelas secara otomatis menggunakan klien HTTP yang sama untuk setiap klien SDK, memungkinkan klien SDK untuk layanan yang berbeda untuk melakukan permintaan HTTP nonblocking. Jika klien SDK tidak menggunakan klien HTTP yang sama, maka permintaan HTTP yang dikirim oleh klien SDK mungkin memblokir orkestrasi janji antar layanan.

Opsi konstruktor klien untuk AWS SDK for PHP Versi 3

Pilihan konstruktor klien dapat disediakan dalam konstruktor klien atau disediakan untuk kelas. `Aws\Sdk` Array opsi yang disediakan untuk jenis klien tertentu dapat bervariasi, berdasarkan klien mana yang Anda buat. Opsi konfigurasi klien kustom ini dijelaskan dalam dokumentasi API setiap klien.

Jika Anda tidak secara eksplisit memberikan opsi konstruktor klien yang dibutuhkan oleh klien, SDK for PHP mencari nilai dari variabel lingkungan atau dari file konfigurasi. AWS Semua klien memerlukan nilai penyedia kredensyal dan Wilayah AWS nilai, jadi Anda harus menetapkan nilai-nilai ini sebagai opsi konstruktor atau eksternal.

Secara default, file konfigurasi yang diperiksa akan berada `.aws/config` di direktori home Anda, biasanya`~/.aws/config`. Namun, Anda dapat menggunakan variabel lingkungan `AWS_CONFIG_FILE` untuk mengatur lokasi file konfigurasi default Anda. Misalnya, ini mungkin berguna jika Anda membatasi akses file ke direktori tertentu dengan. `open_basedir`

Untuk informasi selengkapnya tentang lokasi dan pemformatan `credentials` file bersama AWS `config`, lihat Konfigurasi di Panduan Referensi Alat AWS SDKs dan Konfigurasi.

Untuk detail tentang semua pengaturan konfigurasi global yang dapat Anda atur dalam file AWS konfigurasi atau sebagai variabel lingkungan, lihat Referensi pengaturan konfigurasi dan autentikasi di Panduan Referensi Alat AWS SDKs dan.

Opsi konfigurasi

- api_provider

- credentials
- debug
- statistik
- titik akhir
- endpoint_provider
- endpoint_discovery
- pengurus
- http
- http_handler
- profile
- region
- mencoba lagi
- skema
- layanan
- signature_provider
- signature_version
- ua_append
- gunakan_aws_shared_config_files
- validasi
- versi

Contoh berikut menunjukkan cara meneruskan opsi ke konstruktor klien Amazon S3.

```
use Aws\S3\S3Client;

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

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

Lihat the section called “Membuat klien layanan dasar” untuk informasi lebih lanjut tentang membangun klien.

api_provider

Tipe

`callable`

PHP callable yang menerima argumen tipe, layanan, dan versi, dan mengembalikan array data konfigurasi yang sesuai. Nilai tipe dapat menjadi salah satu dari`api`,`waiter`, atau`paginator`.

Secara default, SDK menggunakan instance `Aws\Api\FileSystemApiProvider` yang memuat file API dari `src/data` folder SDK.

credentials

Tipe

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

Lulus `Aws\Credentials\CredentialsInterface` objek untuk menggunakan instance kredensyal tertentu. Berikut ini menentukan bahwa penyedia kredensi IAM Identity Center harus digunakan. Penyedia ini juga dikenal sebagai penyedia kredensi SSO.

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

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

Jika Anda menggunakan profil bernama, ganti nama profil Anda dengan '`default`' pada contoh sebelumnya. Untuk mempelajari selengkapnya tentang menyiapkan profil bernama, lihat Dibagikan `config` dan `credentials` file di Panduan Referensi Alat AWS SDKs dan Alat.

Jika Anda tidak menentukan penyedia kredensi yang akan digunakan, dan mengandalkan rantai penyedia kredensyal, pesan kesalahan yang dihasilkan dari autentikasi gagal biasanya bersifat generik. Ini dihasilkan dari penyedia terakhir dalam daftar sumber yang diperiksa untuk kredensyal

yang valid, yang mungkin bukan penyedia yang Anda coba gunakan. Saat Anda menentukan penyedia kredensi mana yang akan digunakan, pesan kesalahan apa pun yang dihasilkan akan lebih bermanfaat dan relevan karena hanya dihasilkan dari penyedia itu. Untuk mempelajari lebih lanjut tentang rantai sumber yang diperiksa kredensialnya, lihat Rantai penyedia kredensyal di Panduan Referensi Alat AWS SDKs dan Alat.

Lulus `false` untuk menggunakan kredensyal null dan bukan menandatangani permintaan.

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

Lulus fungsi penyedia kredensyal yang dapat dipanggil untuk membuat kredensyal menggunakan fungsi.

```
use Aws\Credentials\CredentialProvider;

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

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

Teruskan kredensyal yang di-cache ke instance `Aws\CacheInterface` untuk menyimpan nilai yang dikembalikan oleh rantai penyedia default di beberapa proses.

```
use Aws\Credentials\CredentialProvider;
use Aws\PsrCacheAdapter;
use Symfony\Component\Cache\Adapter\FilesystemAdapter;


$cache = new PsrCacheAdapter(new FilesystemAdapter);
$provider = CredentialProvider::defaultProvider();
$cachedProvider = CredentialProvider::cache($provider, $cache);

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

```
]);
```

Anda dapat menemukan informasi lebih lanjut tentang memberikan kredensyal kepada klien di panduan Kredensyal untuk Versi 3. AWS SDK for PHP

> **Note**
>
> Kredensyal dimuat dan divalidasi dengan malas saat digunakan.

debug

Tipe

`bool|array`

Output informasi debug tentang setiap transfer. Informasi debug berisi informasi tentang setiap perubahan status transaksi saat disiapkan dan dikirim melalui kawat. Juga termasuk dalam output debug adalah informasi tentang handler HTTP tertentu yang digunakan oleh klien (misalnya, output cURL debug).

Setel `true` untuk menampilkan informasi debug saat mengirim permintaan.

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

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

Atau, Anda dapat memberikan array asosiatif dengan kunci berikut.

logfn (dapat dipanggil)

Fungsi yang dipanggil dengan pesan log. Secara default, `echo` fungsi PHP digunakan.

stream_size (int)

Ketika ukuran aliran lebih besar dari angka ini, data aliran tidak dicatat. Setel `0` untuk tidak mencatat data aliran apa pun.

scrub_auth (bool)

Setel `false` untuk menonaktifkan penggosokan data autentikasi dari pesan yang dicatat (artinya ID kunci AWS akses dan tanda tangan Anda akan diteruskan ke). `logfn`

http (bool)

Setel `false` untuk menonaktifkan fitur “debug” dari penangan HTTP tingkat rendah (misalnya, keluaran cURL verbose).

auth_headers (array)

Setel ke pemetaan nilai kunci header yang ingin Anda ganti dipetakan ke nilai yang ingin Anda ganti. Nilai-nilai ini tidak digunakan kecuali `scrub_auth` diatur ke`true`.

auth_strings (array)

Setel ke pemetaan kunci-nilai ekspresi reguler untuk dipetakan ke penggantinya. Nilai-nilai ini digunakan oleh scrubber data otentikasi jika `scrub_auth` diatur ke. `true`

```
$s3 = new Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'debug'   => [
        'logfn'        => function ($msg) { echo $msg . "\n"; },
        'stream_size'  => 0,
        'scrub_auth'   => true,
        'http'         => true,
        'auth_headers' => [
            'X-My-Secret-Header' => '[REDACTED]',
        ],
        'auth_strings' => [
            '/SuperSecret=[A-Za-z0-9]{20}/i' => 'SuperSecret=[REDACTED]',
        ],
    ]
]);

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

> **Note**
>
> Opsi ini juga menampilkan informasi handler HTTP yang mendasari yang dihasilkan oleh opsi `http` debug. Output debug sangat berguna saat mendiagnosis masalah di. AWS SDK for

PHP Harap berikan output debug untuk kasus kegagalan terisolasi saat membuka masalah pada SDK.

statistik

Tipe

`bool|array`

Mengikat statistik transfer ke kesalahan dan hasil yang dikembalikan oleh operasi SDK.

Setel `true` untuk mengumpulkan statistik transfer pada permintaan yang dikirim.

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

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

Atau, Anda dapat memberikan array asosiatif dengan kunci berikut.

coba lagi (bool)

Setel `false` untuk menonaktifkan pelaporan pada percobaan ulang yang dicoba. Statistik coba lagi dikumpulkan secara default dan dikembalikan.

http (bool)

Setel `true` untuk mengaktifkan pengumpulan statistik dari adaptor HTTP tingkat rendah (misalnya, nilai yang dikembalikan). GuzzleHttpTransferStats Handler HTTP harus mendukung opsi __on_transfer_stats agar ini memiliki efek. Statistik HTTP dikembalikan sebagai array array asosiatif yang diindeks; setiap array asosiatif berisi statistik transfer yang dikembalikan untuk permintaan oleh handler HTTP klien. Dinonaktifkan secara default.

Jika permintaan dicoba ulang, statistik transfer setiap permintaan dikembalikan, dengan `$result['@metadata']['transferStats']['http'][0]` berisi statistik untuk permintaan

pertama, `$result['@metadata']['transferStats']['http'][1]` berisi statistik untuk permintaan kedua, dan seterusnya.

pengatur waktu (bool)

Setel `true` untuk mengaktifkan pengatur waktu perintah yang melaporkan total waktu jam dinding yang dihabiskan untuk operasi dalam hitungan detik. Dinonaktifkan secara default.

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

// Perform an operation
$result = $s3->listBuckets();
// Inspect the HTTP transfer stats
$stats = $result['@metadata']['transferStats']['http'];
// Inspect the number of retries attempted
$stats = $result['@metadata']['transferStats']['retries_attempted'];
// Inspect the total backoff delay inserted between retries
$stats = $result['@metadata']['transferStats']['total_retry_delay'];
```

titik akhir

Tipe

`string`

URI lengkap dari layanan web. Ini diperlukan untuk layanan, seperti AWS Elemental MediaConvert, yang menggunakan titik akhir khusus akun. Untuk layanan ini, minta titik akhir ini menggunakan `describeEndpoints` metode.

Ini hanya diperlukan saat menghubungkan ke titik akhir kustom (misalnya, versi lokal Amazon S3 atau Amazon DynamoDB Lokal).

Berikut adalah contoh menghubungkan ke Amazon DynamoDB Local:

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

Lihat AWS Wilayah dan Titik Akhir untuk daftar AWS Wilayah dan titik akhir yang tersedia.

endpoint_provider

Tipe

`Aws\EndpointV2\EndpointProviderV2|callable`

Contoh opsional dari EndpointProvider V2 atau PHP callable yang menerima hash opsi, termasuk kunci “layanan” dan “wilayah”. Ini mengembalikan `NULL` atau hash data endpoint, yang kunci “endpoint” diperlukan.

Berikut adalah contoh cara membuat penyedia endpoint minimal.

```
$provider = function (array $params) {
    if ($params['service'] == 'foo') {
        return ['endpoint' => $params['region'] . '.example.com'];
    }
    // Return null when the provider cannot handle the parameters
    return null;
});
```

endpoint_discovery

Tipe

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

Penemuan titik akhir mengidentifikasi dan menghubungkan ke titik akhir yang benar untuk API layanan yang mendukung penemuan titik akhir. Untuk layanan yang mendukung tetapi tidak memerlukan penemuan titik akhir, aktifkan `endpoint_discovery` selama pembuatan klien. Jika layanan tidak mendukung penemuan titik akhir, konfigurasi ini diabaikan.

`Aws\EndpointDiscovery\ConfigurationInterface`

Penyedia konfigurasi opsional yang memungkinkan koneksi otomatis ke titik akhir yang sesuai dari API layanan untuk operasi yang ditentukan oleh layanan.

`Aws\EndpointDiscovery\Configuration`Objek menerima dua opsi, termasuk nilai Boolean, “diaktifkan”, yang menunjukkan apakah penemuan titik akhir diaktifkan, dan bilangan bulat “cache_limit” yang menunjukkan jumlah maksimum kunci dalam cache titik akhir.

Untuk setiap klien yang dibuat, berikan `Aws\EndpointDiscovery\Configuration` objek untuk menggunakan konfigurasi khusus untuk penemuan titik akhir.

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

$enabled = true;
$cache_limit = 1000;

$config = new Aws\EndpointDiscovery\Configuration (
    $enabled,
    $cache_limit
);

$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2',
    'endpoint_discovery' => $config,

]);
```

Berikan instance `Aws\CacheInterface` untuk menyimpan nilai yang dikembalikan oleh penemuan titik akhir di beberapa proses.

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

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

Lewati array ke penemuan titik akhir.

```
use Aws\S3\S3Client;

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

pengurus

Tipe

```
  callable
```

Handler yang menerima objek perintah dan objek permintaan, dan yang mengembalikan promise (`GuzzleHttp\Promise\PromiseInterface`) yang dipenuhi dengan `Aws\ResultInterface` objek atau ditolak dengan. `Aws\Exception\AwsException` Handler tidak menerima handler berikutnya karena terminal dan diharapkan untuk memenuhi perintah. Jika tidak ada handler yang disediakan, handler Guzzle default digunakan.

Anda dapat menggunakan `Aws\MockHandler` untuk mengembalikan hasil yang diejek atau melempar pengecualian tiruan. Anda mengantrekan hasil atau pengecualian, dan MockHandler akan mendeantrekannya dalam urutan FIFO.

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

$mock = new MockHandler();

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

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

```
});

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-east-1',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
$result = $client->listTables();

// This will throw the exception that was enqueued
$client->listTables();
```

http

Tipe

```
array
```

Setel ke array opsi HTTP yang diterapkan ke permintaan HTTP dan transfer yang dibuat oleh SDK.

SDK mendukung opsi konfigurasi berikut:

sertifikat

Tipe

```
string|array
```

Tentukan sertifikat sisi klien yang diformat PEM.

- Tetapkan sebagai string untuk jalur ke hanya file sertifikat.

```
use Aws\S3\S3Client;

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

- Tetapkan sebagai array yang berisi jalur dan kata sandi.

```
use Aws\S3\S3Client;

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

connect_timeout

Float yang menjelaskan jumlah detik untuk menunggu saat mencoba terhubung ke server. Gunakan `0` untuk menunggu tanpa batas (perilaku default).

```
use Aws\DynamoDb\DynamoDbClient;

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

debug

Tipe

`bool|resource`

Menginstruksikan penangan HTTP yang mendasari untuk menampilkan informasi debug. Informasi debug yang disediakan oleh penangan HTTP yang berbeda akan bervariasi.

- Lulus `true` untuk menulis output debug ke STDOUT.
- Teruskan `resource` as yang dikembalikan oleh `fopen` untuk menulis output debug ke sumber daya aliran PHP tertentu.

decode_content

Tipe

`bool`

Menginstruksikan penangan HTTP yang mendasarinya untuk mengembang tubuh respons terkompresi. Saat tidak diaktifkan, badan respons terkompresi mungkin dipompa dengan a. `GuzzleHttp\Psr7\InflateStream`

> **Note**
>
> Penguraian kode konten diaktifkan secara default di handler HTTP default SDK. Untuk alasan kompatibilitas mundur, default ini tidak dapat diubah. Jika Anda menyimpan file terkompresi di Amazon S3, kami sarankan Anda menonaktifkan decoding konten di tingkat klien S3.

```
use Aws\S3\S3Client;
use GuzzleHttp\Psr7\InflateStream;

$client = new S3Client([
    'region'  => 'us-west-2',
    'http'    => ['decode_content' => false],
]);

$result = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'massize_gzipped_file.tgz'
]);

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

delay

Tipe

`int`

Jumlah milidetik yang harus ditunda sebelum mengirim permintaan. Ini sering digunakan untuk menunda sebelum mencoba kembali permintaan.

mengharapkan

Tipe

`bool|string`

Opsi ini diteruskan ke handler HTTP yang mendasarinya. Secara default, header Expect: 100-Continue diatur ketika isi permintaan melebihi 1 MB. `true`atau `false` mengaktifkan atau menonaktifkan header pada semua permintaan. Jika bilangan bulat digunakan, hanya permintaan dengan badan yang melebihi pengaturan ini yang akan menggunakan header. Ketika digunakan sebagai integer, jika ukuran tubuh tidak diketahui header Expect akan dikirim.

> **Warning**
>
> Menonaktifkan header Expect dapat mencegah layanan mengembalikan otentikasi atau kesalahan lainnya. Opsi ini harus dikonfigurasi dengan hati-hati.

kemajuan

Tipe

`callable`

Mendefinisikan fungsi untuk dipanggil ketika kemajuan transfer dibuat. Fungsi menerima argumen berikut:

1. Jumlah total byte yang diharapkan akan diunduh.
2. Jumlah byte yang diunduh sejauh ini.
3. Jumlah byte yang diharapkan akan diunggah.
4. Jumlah byte yang diunggah sejauh ini.

```
use Aws\S3\S3Client;

$client = new S3Client([
```

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

// Apply the http option to a specific command using the "@http"
// command parameter
$result = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'large.mov',
    '@http' => [
        'progress' => function ($expectedDl, $dl, $expectedUl, $ul) {
            printf(
                "%s of %s downloaded, %s of %s uploaded.\n",
                $expectedDl,
                $dl,
                $expectedUl,
                $ul
            );
        }
    ]
]);
```

proxy

Tipe

`string|array`

Anda dapat terhubung ke AWS layanan melalui proxy dengan menggunakan `proxy` opsi.

- Berikan nilai string untuk terhubung ke proxy untuk semua jenis URIs. Nilai string proxy dapat berisi skema, nama pengguna, dan kata sandi. Misalnya, `"http://username:password@192.168.16.1:10"`.
- Berikan array asosiatif pengaturan proxy di mana kuncinya adalah skema URI, dan nilainya adalah proxy untuk URI yang diberikan (yaitu, Anda dapat memberikan proxy yang berbeda untuk titik akhir “http” dan “https”).

```
use Aws\DynamoDb\DynamoDbClient;

// Send requests through a single proxy
$client = new DynamoDbClient([
```

```
    'region'  => 'us-west-2',
    'http'    => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);

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

Anda dapat menggunakan variabel `HTTP_PROXY` lingkungan untuk mengonfigurasi proxy khusus protokol “http”, dan variabel `HTTPS_PROXY` lingkungan untuk mengonfigurasi proxy spesifik “https”.

tenggelam

Tipe

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

`sink`Opsi mengontrol tempat data respons operasi diunduh.

- Berikan `resource` as return by `fopen` untuk mengunduh badan respons ke aliran PHP.
- Berikan jalur ke file pada disk sebagai `string` nilai untuk mengunduh badan respons ke file tertentu pada disk.
- Berikan a `Psr\Http\Message\StreamInterface` untuk mengunduh badan respons ke objek aliran PSR tertentu.

> **Note**
>
> SDK mengunduh badan respons ke aliran temp PHP secara default. Ini berarti bahwa data tetap dalam memori sampai ukuran tubuh mencapai 2 MB, di mana data ditulis ke file sementara pada disk.

sinkron

Tipe

`bool`

`synchronous`Opsi menginformasikan penangan HTTP yang mendasari bahwa Anda bermaksud untuk memblokir hasilnya.

aliran

Tipe

`bool`

Setel `true` untuk memberi tahu penangan HTTP yang mendasarinya bahwa Anda ingin mengalirkan badan respons respons dari layanan web, daripada mengunduh semuanya di muka. Misalnya, opsi ini diandalkan di kelas pembungkus aliran Amazon S3 untuk memastikan bahwa data dialirkan.

batas waktu

Tipe

`float`

Sebuah float yang menjelaskan batas waktu permintaan dalam hitungan detik. Gunakan `0` untuk menunggu tanpa batas (perilaku default).

```
use Aws\DynamoDb\DynamoDbClient;

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

verifikasi

Tipe

```
bool|string
```

Anda dapat menyesuaikan perilaku verifikasi SSL/TLS sertifikat peer SDK menggunakan opsi. `verify` `http`

- Setel `true` untuk mengaktifkan verifikasi sertifikat SSL/TLS peer dan gunakan bundel CA default yang disediakan oleh sistem operasi.
- Setel `false` untuk menonaktifkan verifikasi sertifikat rekan. (Ini tidak aman!)
- Setel ke string untuk menyediakan jalur ke bundel sertifikat CA untuk mengaktifkan verifikasi menggunakan bundel CA kustom.

Jika bundel CA tidak dapat ditemukan untuk sistem Anda dan Anda menerima kesalahan, berikan jalur ke bundel CA ke SDK. Jika Anda tidak memerlukan bundel CA tertentu, Mozilla menyediakan bundel CA yang umum digunakan yang dapat Anda unduh di sini (ini dikelola oleh pengelola cURL). Setelah Anda memiliki bundel CA yang tersedia di disk, Anda dapat mengatur pengaturan `openssl.cafile` PHP .ini untuk menunjuk ke jalur ke file, memungkinkan Anda untuk menghilangkan opsi `verify` permintaan. Anda dapat menemukan lebih banyak detail tentang sertifikat SSL di situs web cURL.

```
use Aws\DynamoDb\DynamoDbClient;

// Use a custom CA bundle
$client = new DynamoDbClient([
    'region'  => 'us-west-2',
    'http'    => [
        'verify' => '/path/to/my/cert.pem'
    ]
]);

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

```
]);
```

http_handler

Tipe

```
callable
```

`http_handler`Opsi ini digunakan untuk mengintegrasikan SDK dengan klien HTTP lainnya. `http_handler`Opsi adalah fungsi yang menerima `Psr\Http\Message\RequestInterface` objek dan array `http` opsi yang diterapkan pada perintah, dan mengembalikan `GuzzleHttp\Promise\PromiseInterface` objek yang dipenuhi dengan `Psr\Http\Message\ResponseInterface` objek atau ditolak dengan array data pengecualian berikut:

- `exception`- (`\Exception`) pengecualian yang ditemui.
- `response`- (`Psr\Http\Message\ResponseInterface`) tanggapan yang diterima (jika ada).
- `connection_error`- (bool) diatur `true` untuk menandai kesalahan sebagai kesalahan koneksi. Menyetel nilai ini `true` juga memungkinkan SDK untuk mencoba kembali operasi secara otomatis, jika diperlukan.

SDK secara otomatis mengubah yang diberikan `http_handler` menjadi `handler` opsi normal dengan membungkus yang disediakan `http_handler` dengan objek. `Aws\WrappedHttpHandler`

Secara default, SDK menggunakan Guzzle sebagai handler HTTP-nya. Anda dapat menyediakan handler HTTP yang berbeda di sini, atau menyediakan klien Guzzle dengan opsi yang ditentukan khusus Anda sendiri.

Mengatur versi TLS

Salah satu kasus penggunaan adalah mengatur versi TLS yang digunakan oleh Guzzle dengan Curl, dengan asumsi Curl diinstal di lingkungan Anda. Perhatikan batasan versi Curl untuk versi TLS apa yang didukung. Secara default, versi terbaru digunakan. Jika versi TLS ditetapkan secara eksplisit, dan server jarak jauh tidak mendukung versi ini, itu akan menghasilkan kesalahan alih-alih menggunakan versi TLS sebelumnya.

Anda dapat menentukan versi TLS yang digunakan untuk operasi klien tertentu dengan mengatur opsi `debug` klien ke true dan memeriksa output koneksi SSL. Baris itu mungkin terlihat seperti: `SSL connection using TLSv1.2`

Contoh pengaturan TLS 1.2 dengan Guzzle 6:

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

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

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

> **Note**
>
> `http_handler`Opsi ini menggantikan opsi apa pun yang disediakan`handler`.

profile

Tipe

`string`

Opsi “profil” menentukan profil mana yang akan digunakan saat kredensyal dibuat dari file AWS kredensyal di direktori HOME Anda (biasanya). `~/.aws/credentials` Pengaturan ini mengesampingkan variabel `AWS_PROFILE` lingkungan.

> **Note**
>
> Saat Anda menentukan opsi “profil”, `"credentials"` opsi diabaikan dan pengaturan terkait kredensyal dalam file AWS konfigurasi (biasanya`~/.aws/config`) diabaikan.

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

Lihat Kredensyal untuk AWS SDK for PHP Versi 3 untuk informasi selengkapnya tentang mengonfigurasi kredensyal dan format file.ini.

region

Tipe

`string`

Wajib

true

AWS Wilayah untuk terhubung ke. Lihat AWS Wilayah dan Titik Akhir untuk daftar Wilayah yang tersedia.

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

mencoba lagi

Tipe

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

Default

`int(3)`

Mengkonfigurasi mode coba lagi dan jumlah maksimum percobaan ulang yang diizinkan untuk klien. Lulus `0` untuk menonaktifkan percobaan ulang.

Tiga mode coba lagi adalah:

- `legacy`- implementasi coba lagi warisan default
- `standard`- menambahkan sistem kuota coba lagi untuk mencegah percobaan ulang yang tidak mungkin berhasil
- `adaptive`- dibangun pada mode standar, menambahkan pembatas tingkat sisi klien. Perhatikan mode ini dianggap eksperimental.

Konfigurasi untuk percobaan ulang terdiri dari mode dan upaya maksimal yang akan digunakan untuk setiap permintaan. Konfigurasi dapat diatur di beberapa lokasi yang berbeda, dalam urutan prioritas berikut.

Urutan Prioritas

Urutan prioritas untuk konfigurasi coba lagi adalah sebagai berikut (1 mengesampingkan 2-3, dll.):

1. Opsi konfigurasi klien
2. Variabel-variabel lingkungan
3. AWS File konfigurasi bersama

Variabel lingkungan

- `AWS_RETRY_MODE`- diatur k`elegacy`,`standard`, atau`adaptive`.
- `AWS_MAX_ATTEMPTS`- diatur ke nilai integer untuk upaya maksimal per permintaan

Kunci file konfigurasi bersama

- `retry_mode`- diatur k`elegacy`,`standard`, atau`adaptive`.
- `max_attempts`- diatur ke nilai integer untuk upaya maksimal per permintaan

Konfigurasi klien

Contoh berikut menonaktifkan percobaan ulang untuk klien Amazon DynamoDB.

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

```
    'region'  => 'us-west-2',
    'retries' => 0
]);
```

Contoh berikut melewati integer, yang akan default ke `legacy` mode dengan diteruskan dalam jumlah percobaan ulang

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

`Aws\Retry\Configuration`Objek menerima dua parameter, mode coba lagi

dan bilangan bulat untuk upaya maksimum per permintaan. Contoh ini diteruskan dalam

`Aws\Retry\Configuration`objek untuk konfigurasi coba lagi.

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

$enabled = true;
$cache_limit = 1000;

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

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

Contoh ini masuk dalam array untuk konfigurasi coba lagi.

```
use Aws\S3\S3Client;

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

```
        'max_attempts' => 7
    ],
]);
```

Contoh ini meneruskan instance `Aws\CacheInterface` untuk menyimpan nilai yang dikembalikan oleh penyedia konfigurasi coba lagi default.

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

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

skema

Tipe

`string`

Default

`string(5) "https"`

Skema URI untuk digunakan saat menghubungkan. SDK menggunakan titik akhir “https” (yaitu, menggunakan koneksi SSL/TLS) secara default. Anda dapat mencoba menghubungkan ke layanan melalui titik akhir “http” yang tidak terenkripsi dengan menyetel `scheme` ke “http”.

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

Lihat AWS Wilayah dan Titik Akhir untuk daftar titik akhir dan apakah layanan mendukung skema.
`http`

layanan

Tipe

`string`

Wajib

true

Nama layanan yang akan digunakan. Nilai ini diberikan secara default saat menggunakan klien yang disediakan oleh SDK (yaitu,`Aws\S3\S3Client`). Opsi ini berguna saat menguji layanan yang belum dipublikasikan di SDK tetapi Anda telah tersedia di disk.

signature_provider

Tipe

`callable`

Callable yang menerima nama versi tanda tangan (misalnya,`v4`), nama layanan, dan AWS Wilayah dan menampilkan `Aws\Signature\SignatureInterface` objek atau `NULL` jika penyedia dapat membuat tanda tangan untuk parameter yang diberikan. Penyedia ini digunakan untuk membuat penandatangan yang digunakan oleh klien.

Ada berbagai fungsi yang disediakan oleh SDK di `Aws\Signature\SignatureProvider` kelas yang dapat digunakan untuk membuat penyedia tanda tangan yang disesuaikan.

signature_version

Tipe

`string`

String yang mewakili versi tanda tangan khusus untuk digunakan dengan layanan (misalnya,`v4`, dll.). Versi tanda tangan per operasi MUNGKIN mengganti versi tanda tangan yang diminta ini, jika diperlukan.

Contoh berikut menunjukkan cara mengonfigurasi klien Amazon S3 untuk menggunakan tanda tangan versi 4:

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

> **Note**
>
> Yang `signature_provider` digunakan oleh klien Anda HARUS dapat membuat `signature_version` opsi yang Anda berikan. Default yang `signature_provider` digunakan oleh SDK dapat membuat objek tanda tangan untuk versi tanda tangan “v4" dan “anonim”.

ua_append

Tipe

`string|string[]`

Default

`[]`

Sebuah string atau array string yang ditambahkan ke string user-agent diteruskan ke handler HTTP.

gunakan_aws_shared_config_files

Tipe

`bool|array`

Default

`bool(true)`

Setel ke false untuk menonaktifkan pemeriksaan file konfigurasi bersama di '~/. aws/config’ and ‘~/.aws/credentials'. Ini akan mengesampingkan variabel AWS_CONFIG_FILE lingkungan.

validasi

Tipe

`bool|array`

Default

`bool(true)`

Setel `false` untuk menonaktifkan validasi parameter sisi klien. Anda mungkin menemukan bahwa mematikan validasi akan sedikit meningkatkan kinerja klien, tetapi perbedaannya dapat diabaikan.

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

Setel ke array asosiatif opsi validasi untuk mengaktifkan kendala validasi tertentu:

- `required`- Validasi bahwa parameter yang diperlukan ada (aktif secara default).
- `min`- Validasi panjang minimum nilai (aktif secara default).
- `max`- Validasi panjang maksimum suatu nilai.
- `pattern`- Validasi bahwa nilai cocok dengan ekspresi reguler.

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

versi

Tipe

`string`

Wajib

false

Opsi ini menentukan versi layanan web yang akan digunakan (misalnya,`2006-03-01`).

Dimulai dengan versi 3.277.10 SDK, opsi “versi” tidak diperlukan. Jika Anda tidak menentukan opsi “versi”, SDK menggunakan versi terbaru klien layanan.

Dua situasi memerlukan parameter “versi” saat Anda membuat klien layanan.

- Anda menggunakan versi PHP SDK lebih awal dari 3.277.10.
- Anda menggunakan versi 3.277.10 atau yang lebih baru dan ingin menggunakan versi selain 'terbaru' untuk klien layanan.

Misalnya, cuplikan berikut menggunakan SDK versi 3.279.7, tetapi bukan versi terbaru untuk versi. `Ec2Client`

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

Menentukan batasan versi memastikan bahwa kode Anda tidak akan terpengaruh oleh perubahan yang melanggar yang dibuat pada layanan.

Daftar versi API yang tersedia dapat ditemukan di halaman dokumentasi API setiap klien. Jika Anda tidak dapat memuat versi API tertentu, Anda mungkin perlu memperbarui salinan SDK.

Mengatur Wilayah AWS untuk AWS SDK for PHP Versi 3

Klien SDK terhubung ke Layanan AWS dalam spesifik Wilayah AWS yang Anda tentukan saat Anda membuat klien. Konfigurasi ini memungkinkan aplikasi Anda berinteraksi dengan AWS sumber daya di wilayah geografis tersebut. Saat Anda membuat klien layanan tanpa menyetel Region secara eksplisit, SDK menggunakan Region default dari konfigurasi eksternal Anda.

Rantai resolusi wilayah

AWS SDK for PHP Versi 3 menggunakan urutan berikut untuk menentukan Wilayah mana yang digunakan klien layanan:

1. Wilayah disediakan dalam kode—Jika Anda secara eksplisit menyetel Wilayah dalam opsi konstruktor klien, ini lebih diutamakan daripada semua sumber lainnya.

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

2. Variabel lingkungan—Jika tidak ada Wilayah yang disediakan dalam kode, SDK akan memeriksa variabel lingkungan ini secara berurutan:
   - `AWS_REGION`
   - `AWS_DEFAULT_REGION`

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

3. AWS file konfigurasi—Jika tidak ada variabel lingkungan Region yang disetel, SDK akan memeriksa file konfigurasi: AWS
   a. SDK terlihat `~/.aws/config` (atau lokasi yang ditentukan oleh variabel `AWS_CONFIG_FILE` lingkungan)
   b. SDK memeriksa setelan wilayah dalam profil yang ditentukan oleh variabel lingkungan `AWS_PROFILE`
   c. Jika tidak `AWS_PROFILE` ditentukan, SDK menggunakan profil “default”

   Sebagai contoh, asumsikan kita memiliki pengaturan file konfigurasi berikut:

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

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

Jika variabel `AWS_PROFILE` lingkungan diatur dengan nilai “produksi”, klien menggunakan`eu-central-1 Region`. Jika tidak ada variabel `AWS_PROFILE` lingkungan, klien menggunakan `eu-west-1` Region.

4. Jika SDK tidak menemukan nilai Region di salah satu sumber di atas, SDK akan menampilkan pengecualian karena nilai Region adalah setelan wajib untuk klien layanan.

Praktik terbaik

Pertimbangkan praktik terbaik berikut saat bekerja dengan Wilayah di AWS SDK for PHP Versi 3:

Secara eksplisit mengatur Wilayah dalam kode produksi

Untuk aplikasi produksi, kami sarankan secara eksplisit mengatur Wilayah dalam kode Anda daripada mengandalkan variabel lingkungan atau. `config` Ini membuat kode Anda lebih dapat diprediksi dan kurang bergantung pada konfigurasi eksternal.

Gunakan variabel lingkungan untuk pengembangan dan pengujian

Untuk lingkungan pengembangan dan pengujian, menggunakan variabel lingkungan memungkinkan lebih banyak fleksibilitas tanpa mengubah kode.

Gunakan profil untuk beberapa lingkungan

Jika aplikasi Anda perlu bekerja dengan beberapa AWS lingkungan, pertimbangkan untuk menggunakan profil yang berbeda dalam AWS `config` file Anda dan beralih di antara mereka sesuai kebutuhan.

Menggunakan penyedia kredensi AWS SDK for PHP Versi 3

Untuk informasi referensi tentang mekanisme kredensional yang tersedia AWS SDKs, lihat Kredensial dan akses di Panduan Referensi Alat AWS SDKs dan Alat.

> **Important**
>
> Untuk keamanan, kami sangat menyarankan Anda untuk tidak menggunakan akun root untuk AWS akses. Selalu lihat praktik terbaik Keamanan di IAM di Panduan Pengguna IAM untuk rekomendasi keamanan terbaru.

Peran penyedia kredensial di AWS SDK for PHP Versi 3 adalah untuk mencari dan menyediakan kredensil ke klien SDK. Layanan AWS SDK menggunakan kredensil yang disumberkannya untuk mengautentikasi dengan layanan dengan menandatangani setiap permintaan secara kriptografis. Kredensil biasanya terdiri dari kunci akses—ID kunci akses dan kunci akses rahasia bersama-sama.

Saat Anda menggunakan kredensil sementara, seperti saat Anda menyiapkan autentikasi Pusat Identitas IAM atau mengonfigurasi runtime Anda untuk mengambil peran IAM, token sesi ditambahkan ke kunci akses, menyediakan akses terbatas waktu ke sumber daya. AWS

Apa yang dimaksud dengan penyedia kredensi di AWS SDK for PHP versi 3?

Penyedia kredensi adalah fungsi yang mengembalikan a `GuzzleHttp\Promise\PromiseInterface` yang dipenuhi dengan `Aws\Credentials\CredentialsInterface` instance atau ditolak dengan. `Aws\Exception\CredentialsException` SDK menyediakan beberapa implementasi fungsi penyedia kredensi atau Anda dapat menerapkan logika kustom Anda sendiri untuk membuat kredensil atau untuk mengoptimalkan pemuatan kredenal.

Penyedia kredensi diteruskan ke opsi konstruktor `credentials` klien. Penyedia kredensi bersifat asinkron, yang memaksa mereka untuk dievaluasi secara malas setiap kali operasi API dipanggil. Dengan demikian, meneruskan fungsi penyedia kredensi ke konstruktor klien SDK tidak segera memvalidasi kredensialnya. Jika penyedia kredensi tidak mengembalikan objek kredensial, operasi API akan ditolak dengan file. `Aws\Exception\CredentialsException`

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

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

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

Memahami rantai penyedia kredensi default di AWS SDK for PHP Versi 3

Rantai penyedia kredenal default terdiri dari serangkaian penyedia kredensi bawaan yang dipanggil SDK. Ini diimplementasikan oleh fungsi penyedia kredensi DefaultProvider tanpa parameter. Setelah kredensi yang valid ditemukan, pencarian dihentikan.

Penyedia kredensi AWS SDK for PHP mengeksekusi dengan urutan sebagai berikut:

- **env**provider - SDK mencari kunci AWS akses yang telah ditetapkan sebagai variabel lingkungan.
- **assumeRoleWithWebIdentityCredentialProvider**penyedia - SDK mencari peran IAM dan pengaturan file token identitas web.
- Pada titik ini dalam rantai, SDK mencari konfigurasi dalam file bersama AWS `config` dan `credentials` file. SDK mencari konfigurasi di bawah profil “default”, tetapi jika variabel `AWS_PROFILE` lingkungan disetel, SDK menggunakan nilai profil bernama.
  - **sso**penyedia - SDK mencari pengaturan konfigurasi Pusat Identitas IAM di file bersama`config`.
  - **login provider**- SDK mencari pengaturan konfigurasi sesi login AWS konsol di `config` file bersama.
  - **process**penyedia - SDK mencari `credential_process` pengaturan dalam `credentials` file bersama.
  - **ini**penyedia - SDK mencari AWS kredensi atau informasi peran IAM dalam file bersama. `credentials`
  - **process**penyedia - SDK mencari `credential_process` pengaturan dalam `config` file bersama.
  - **ini**penyedia - SDK mencari AWS kredensi atau informasi peran IAM dalam file bersama. `config`
- **ecsCredentials**penyedia - SDK mencari variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` atau `AWS_CONTAINER_CREDENTIALS_FULL_URI` yang memberikan informasi untuk memperoleh kredensi sementara.
- **instanceProfile**penyedia - SDK menggunakan layanan Metadata EC2 Instance untuk mendapatkan peran IAM yang ditentukan dalam profil instance. Dengan menggunakan informasi peran, SDK memperoleh kredensi sementara.

> **Note**
>
> Hasil dari penyedia default secara otomatis memoisasi.

Anda dapat meninjau kode untuk rantai dalam kode GitHub sumber.

Penyedia kredensi bawaan di AWS SDK for PHP Versi 3

SDK menyediakan beberapa penyedia kredensi bawaan yang dapat Anda gunakan secara individual atau gabungkan dalam rantai penyedia kredensi khusus.

Saat Anda menentukan penyedia kredensi selama pembuatan klien layanan, SDK mencoba memuat kredensi hanya dengan menggunakan penyedia kredensi yang ditentukan. Itu tidak menggunakan rantai penyedia kredensi default. Jika Anda tahu bahwa Anda ingin klien layanan menggunakan `instanceProfile` penyedia, Anda dapat melakukan hubungan pendek rantai default dengan menentukan `instanceProfile` penyedia di konstruktor klien layanan:

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

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

$client = new S3Client([
    'region'      => 'us-west-2',
    'credentials' => $memoizedProvider  // The default credential provider chain is not
 used.
]);
```

> **Important**
>
> Penyedia kredensi dipanggil setiap kali operasi API dilakukan. Jika memuat kredensi adalah tugas yang mahal (misalnya, memuat dari disk atau sumber daya jaringan), atau jika kredenal tidak di-cache oleh penyedia Anda, pertimbangkan untuk membungkus penyedia kredensi Anda dalam suatu fungsi. `Aws\Credentials\CredentialProvider::memoize` Penyedia kredensi default yang digunakan oleh SDK secara otomatis memoisasi.

Topik

- loginpenyedia di SDK for PHP
- assumeRolepenyedia di SDK for PHP
- ssopenyedia di SDK for PHP
- defaultProviderpenyedia di SDK for PHP
- ecsCredentialspenyedia di SDK for PHP
- envpenyedia di SDK for PHP
- assumeRoleWithWebIdentityCredentialProviderpenyedia di SDK for PHP
- inipenyedia di SDK for PHP
- processpenyedia di SDK for PHP
- instanceProfilepenyedia di SDK for PHP

loginpenyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::login`mencoba memuat kredenal yang dikonfigurasi oleh sesi login berbasis browser yang difasilitasi oleh alat seperti CLI. AWS Setelah otentikasi, buat AWS kredenal sementara yang berfungsi di seluruh lokal AWS SDKs dan alat.

Dengan proses ini, Anda dapat mengautentikasi menggunakan kredensi root yang dibuat selama pengaturan akun awal, pengguna IAM, atau identitas federasi dari penyedia identitas Anda, dan SDK for AWS PHP secara otomatis mengelola kredensi sementara untuk Anda. Pendekatan ini meningkatkan keamanan dengan menghilangkan kebutuhan untuk menyimpan kredensi jangka panjang secara lokal.

Ketika Anda menjalankan `aws login` perintah, Anda dapat memilih dari sesi konsol aktif Anda, atau masuk melalui alur otentikasi berbasis browser dan ini akan secara otomatis menghasilkan kredensi sementara. AWS SDK for PHP akan secara otomatis me-refresh kredenal ini, menggunakan layanan Sign-In, hingga 12 jam.

Penyedia login mencoba memuat token akses yang dihasilkan oleh alur kerja sesi login yang disebutkan sebelumnya, berdasarkan profil yang disediakan. Jika tidak ada profil yang disediakan saat memanggil penyedia, itu akan mencoba menyelesaikan profil dengan terlebih dahulu memeriksa variabel `AWS_PROFILE` lingkungan, sebelum kembali ke profil`default`. Konfigurasi dalam kode dapat diteruskan ke penyedia, di mana ia akan mencari `region` nilai untuk klien layanan Masuk yang digunakan untuk menyegarkan kredensional. Jika tidak ada wilayah yang disediakan dalam

larik konfigurasi, penyedia akan mencoba menyelesaikan wilayah dengan memeriksa variabel `AWS_REGION` lingkungan, lalu nilai wilayah ditetapkan dalam profil yang diselesaikan. Jika tidak ada wilayah yang dapat ditemukan, penyedia akan mengembalikan janji yang ditolak dengan instruksi tentang cara mengonfigurasi wilayah.

Penyedia disebut sebagai bagian dari rantai default dan dapat dipanggil secara langsung.

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

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

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

Secara default, jika tidak ada konfigurasi kredensional yang disediakan pada klien layanan yang ingin Anda gunakan, penyedia ini akan dipanggil sebagai bagian dari rantai `defaultProvider()` kredensial. Dalam skenario ini, wilayah klien layanan secara otomatis diteruskan ke `login()` penyedia. Juga dalam skenario ini, nilai profil yang diteruskan ke penyedia login akan diselesaikan dengan memeriksa variabel `AWS_PROFILE` lingkungan, sebelum kembali ke profil`default`.

`assumeRole`penyedia di SDK for PHP

Jika Anda menggunakan `Aws\Credentials\AssumeRoleCredentialProvider` untuk membuat kredensional dengan mengambil peran, Anda perlu memberikan `'client'` informasi dengan `StsClient` objek dan `'assume_role_params'` detail, seperti yang ditunjukkan.

> **Note**
>
> Untuk menghindari pengambilan AWS STS kredensi yang tidak perlu pada setiap operasi API, Anda dapat menggunakan `memoize` fungsi tersebut untuk menangani penyegaran kredenal secara otomatis saat kredensialnya kedaluwarsa. Lihat kode berikut sebagai contoh.

```
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
]);
```

Untuk informasi lebih lanjut mengenai`'assume_role_params'`, lihat AssumeRole.

ssopenyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::sso`adalah penyedia kredensi masuk tunggal. Penyedia ini juga dikenal sebagai penyedia AWS IAM Identity Center kredensi.

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

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

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

Jika Anda menggunakan profil bernama, ganti nama profil Anda dengan '`default`' pada contoh sebelumnya. Untuk mempelajari selengkapnya tentang menyiapkan profil bernama, lihat Dibagikan `config` dan `credentials` file di Panduan Referensi Alat AWS SDKs dan Alat. Atau, Anda dapat menggunakan variabel `AWS_PROFILE`lingkungan untuk menentukan pengaturan profil mana yang akan digunakan.

Untuk memahami lebih lanjut cara kerja penyedia Pusat Identitas IAM, lihat Memahami autentikasi Pusat Identitas IAM di Panduan Referensi Alat AWS SDKs dan Alat.

`defaultProvider`penyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::defaultProvider`adalah penyedia kredensi default dan juga disebut rantai penyedia kredensi default. Penyedia ini digunakan jika Anda menghilangkan `credentials` opsi saat membuat klien. Misalnya, jika Anda membuat S3Client seperti yang ditunjukkan pada cuplikan berikut, SDK akan menggunakan penyedia default:

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

Anda juga dapat menggunakan DefaultProvider dalam kode jika Anda ingin menyediakan parameter ke penyedia kredensi tertentu dalam rantai. Misalnya contoh berikut menyediakan batas waktu koneksi kustom dan pengaturan coba lagi jika fungsi `ecsCredentials` penyedia digunakan.

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

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

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

ecsCredentialspenyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::ecsCredentials`mencoba memuat kredensi dengan `GET` permintaan, yang URI-nya ditentukan oleh variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` dalam wadah.

```
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
]);
```

envpenyedia di SDK for PHP

Menggunakan variabel lingkungan untuk memuat kredensi Anda mencegah Anda berbagi kunci akses AWS rahasia secara tidak sengaja. Kami menyarankan Anda untuk tidak pernah menambahkan kunci AWS akses Anda langsung ke klien dalam file produksi apa pun.

Untuk mengautentikasi ke Amazon Web Services, SDK terlebih dahulu memeriksa kredensi di variabel lingkungan Anda. SDK menggunakan `getenv()` fungsi untuk mencari variabel`AWS_ACCESS_KEY_ID`,`AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` lingkungan. Kredensial ini disebut sebagai kredensial lingkungan. Untuk petunjuk tentang cara mendapatkan nilai ini, lihat Mengautentikasi menggunakan kredensi jangka pendek di Panduan Referensi Alat AWS SDKs dan Alat.

Jika Anda meng-hosting aplikasi AWS Elastic Beanstalk, Anda dapat mengatur variabel`AWS_ACCESS_KEY_ID`,`AWS_SECRET_KEY`, dan `AWS_SESSION_TOKEN` lingkungan melalui AWS Elastic Beanstalk konsol sehingga SDK dapat menggunakan kredenal tersebut secara otomatis.

Untuk informasi selengkapnya tentang cara mengatur variabel lingkungan, lihat Dukungan variabel lingkungan di Panduan Referensi Alat AWS SDKs dan Alat. Juga, untuk daftar semua variabel lingkungan yang didukung oleh sebagian besar AWS SDKs, lihat Daftar variabel lingkungan.

Anda juga dapat mengatur variabel lingkungan di baris perintah, seperti yang ditunjukkan di sini.

Linux

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

Windows

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

`Aws\Credentials\CredentialProvider::env`mencoba memuat kredensi dari variabel lingkungan.

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

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

`assumeRoleWithWebIdentityCredentialProvider`penyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider`mencoba memuat kredensional dengan mengasumsikan peran. Jika variabel lingkungan `AWS_ROLE_ARN` dan `AWS_WEB_IDENTITY_TOKEN_FILE` ada, penyedia akan mencoba untuk mengambil peran yang ditentukan saat `AWS_ROLE_ARN` menggunakan token pada disk pada jalur lengkap yang ditentukan dalam`AWS_WEB_IDENTITY_TOKEN_FILE`. Jika variabel lingkungan digunakan, penyedia akan mencoba untuk mengatur sesi dari variabel `AWS_ROLE_SESSION_NAME` lingkungan.

Jika variabel lingkungan tidak disetel, penyedia akan menggunakan profil default, atau yang ditetapkan sebagai`AWS_PROFILE`. Penyedia membaca profil dari `~/.aws/credentials` dan secara `~/.aws/config` default, dan dapat membaca dari profil yang ditentukan dalam opsi `filename` konfigurasi. Penyedia akan mengambil peran dalam `role_arn` profil, membaca token dari jalur lengkap yang ditetapkan`web_identity_token_file`. `role_session_name`akan digunakan jika diatur pada profil.

Penyedia disebut sebagai bagian dari rantai default dan dapat dipanggil secara langsung.

```
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
]);
```

Secara default, penyedia kredensi ini akan mewarisi wilayah yang dikonfigurasi yang akan digunakan oleh StsClient untuk mengambil peran. Secara opsional, penuh StsClient dapat disediakan. Kredensi harus ditetapkan seperti `false` pada apa pun yang disediakan. StsClient

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

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

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

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

inipenyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::ini`mencoba memuat kredensional dari file bersama `config` dan`credentials`. Secara default, SDK mencoba memuat profil “default” dari AWS `credentials` file bersama yang terletak di`~/.aws/credentials`. Jika SDK menemukan variabel `AWS_SDK_LOAD_NONDEFAULT_CONFIG` lingkungan, SDK juga memeriksa profil “default” dalam AWS `config` file bersama yang terletak di`~/.aws/config`.

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

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

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

Anda dapat menggunakan profil kustom atau lokasi file.ini dengan memberikan argumen ke fungsi yang membuat penyedia.

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::ini($profile, $path);
$provider = CredentialProvider::memoize($provider);

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

`process`penyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::process`mencoba memuat kredensi dengan mengeksekusi `credential_process` nilai yang ditentukan dalam profil dalam file konfigurasi bersama AWS.

Secara default, SDK mencoba memuat profil “default” terlebih dahulu dari AWS `credentials` file bersama yang terletak di`~/.aws/credentials`. Jika profil “default” tidak ditemukan di `credentials` file bersama, SDK akan terlihat di `config` file bersama untuk profil default. Berikut ini adalah contoh konfigurasi untuk `credentials` file bersama.

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

SDK akan memanggil `credential_process` perintah persis seperti yang diberikan dengan menggunakan `shell_exec` fungsi PHP dan kemudian membaca data JSON dari stdout. `credential_process`Harus menulis kredensi ke stdout dalam format berikut:

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

```
}
```

`SessionToken`dan `Expiration` bersifat opsional. Jika ada, kredensialnya akan diperlakukan sebagai sementara.

```
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
]);
```

Anda dapat menggunakan profil kustom atau lokasi file.ini dengan memberikan argumen ke fungsi yang membuat penyedia.

```
$profile = 'production';
$path = '/full/path/to/credentials.ini';

$provider = CredentialProvider::process($profile, $path);
$provider = CredentialProvider::memoize($provider);

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

`instanceProfile`penyedia di SDK for PHP

`Aws\Credentials\CredentialProvider::instanceProfile`mencoba memuat kredensi untuk peran IAM yang ditentukan dalam profil instans 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
]);
```

Secara default, penyedia mencoba lagi mengambil kredensi hingga tiga kali. Jumlah percobaan ulang dapat diatur dengan `retries` opsi, dan dinonaktifkan sepenuhnya dengan mengatur opsi `0` seperti yang ditunjukkan pada kode berikut.

```
use Aws\Credentials\CredentialProvider;

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

Jika variabel lingkungan `AWS_METADATA_SERVICE_NUM_ATTEMPTS` tersedia, nilainya lebih diutamakan daripada opsi 'coba ulang' yang ditunjukkan sebelumnya.

> **Note**
>
> Anda dapat menonaktifkan upaya pemuatan ini dari profil EC2 instans Amazon dengan menyetel variabel `AWS_EC2_METADATA_DISABLED` lingkungan ke`true`.

Merantai penyedia kredensi di SDK for PHP

Anda dapat menghubungkan penyedia kredensi dengan menggunakan `Aws\Credentials\CredentialProvider::chain()` fungsi tersebut. Fungsi ini menerima sejumlah argumen variadik, yang masing-masing merupakan fungsi penyedia kredensi. Fungsi ini kemudian mengembalikan fungsi baru yang merupakan komposisi fungsi yang disediakan, sehingga mereka dipanggil satu demi satu sampai salah satu penyedia mengembalikan janji yang berhasil dipenuhi.

`defaultProvider`Menggunakan komposisi ini untuk memeriksa beberapa penyedia sebelum gagal. Sumber `defaultProvider` menunjukkan penggunaan `chain` fungsi.

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

Membuat penyedia kredensi khusus untuk digunakan dengan SDK for PHP

Penyedia kredensi hanyalah fungsi yang ketika dipanggil mengembalikan promise (`GuzzleHttp\Promise\PromiseInterface`) yang dipenuhi dengan `Aws\Credentials\CredentialsInterface` objek atau ditolak dengan. `Aws\Exception\CredentialsException`

Praktik terbaik untuk membuat penyedia adalah membuat fungsi yang dipanggil untuk membuat penyedia kredensi yang sebenarnya. Sebagai contoh, inilah sumber `env` penyedia (sedikit dimodifikasi untuk tujuan contoh). Perhatikan bahwa itu adalah fungsi yang mengembalikan fungsi penyedia yang sebenarnya. Ini memungkinkan Anda untuk dengan mudah membuat penyedia kredensi dan menyebarkannya sebagai nilai.

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

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

```
                new Credentials($key, $secret, getenv(self::ENV_SESSION))
            );
        }

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

Memoisasi kredensi di SDK for PHP

Terkadang Anda mungkin perlu membuat penyedia kredensi yang mengingat nilai pengembalian sebelumnya. Ini dapat berguna untuk kinerja saat memuat kredensil adalah operasi yang mahal atau saat menggunakan `Aws\Sdk` kelas untuk berbagi penyedia kredensi di beberapa klien. Anda dapat menambahkan memoisasi ke penyedia kredensi dengan membungkus fungsi penyedia kredensi dalam suatu fungsi. `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());
```

Saat kredensil yang dimemoisasi kedaluwarsa, pembungkus memoize memanggil penyedia yang dibungkus dalam upaya untuk menyegarkan kredensil.

Asumsikan peran IAM menggunakan AWS SDK for PHP Versi 3

Menggunakan peran IAM untuk kredenal variabel EC2 instans Amazon

Jika Anda menjalankan aplikasi di EC2 instans Amazon, cara yang lebih disukai untuk memberikan kredensi untuk melakukan panggilan AWS adalah dengan menggunakan peran IAM untuk mendapatkan kredenal keamanan sementara.

Saat Anda menggunakan peran IAM, Anda tidak perlu khawatir tentang manajemen kredensi dari aplikasi Anda. Mereka mengizinkan instance untuk “mengambil” peran dengan mengambil kredensi sementara dari server metadata EC2 instans Amazon.

Kredensial sementara, sering disebut sebagai kredensial profil instance, memungkinkan akses ke tindakan dan sumber daya yang diizinkan oleh kebijakan peran tersebut. Amazon EC2 menangani semua kerja keras untuk mengautentikasi instans secara aman ke layanan IAM untuk mengambil peran, dan menyegarkan kredenal peran yang diambil secara berkala. Ini membuat aplikasi Anda aman dengan hampir tidak ada pekerjaan di pihak Anda. Untuk daftar layanan yang menerima kredensi keamanan sementara, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

> **Note**
>
> Untuk menghindari menekan layanan metadata setiap saat, Anda dapat meneruskan instance `Aws\CacheInterface` in sebagai `'credentials'` opsi ke konstruktor klien. Ini memungkinkan SDK menggunakan kredenal profil instans cache sebagai gantinya. Untuk detailnya, lihat Konfigurasi untuk AWS SDK for PHP Versi 3.

Untuk informasi selengkapnya tentang mengembangkan EC2 aplikasi Amazon menggunakan SDKs, lihat Menggunakan peran IAM untuk EC2 instans Amazon di Panduan Referensi Alat AWS SDKs dan Alat.

Membuat dan menetapkan peran IAM ke instans Amazon EC2

1. Buat klien IAM.

   Impor

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

```
use Aws\Iam\IamClient;
```

Kode Sampel

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

2. Buat peran IAM dengan izin untuk tindakan dan sumber daya yang akan Anda gunakan.

   Kode Sampel

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

3. Buat profil instans IAM dan simpan Amazon Resource Name (ARN) dari hasilnya.

   > **Note**
   >
   > Jika Anda menggunakan konsol IAM alih-alih AWS SDK for PHP, konsol akan membuat profil instance secara otomatis dan memberinya nama yang sama dengan peran yang sesuai dengannya.
   >
   > Kode Sampel

```
$IPN = 'InstanceProfileName';

$result = $client->createInstanceProfile([
    'InstanceProfileName' => $IPN ,
]);

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

4. Buat EC2 klien Amazon.

   Impor

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

use Aws\Ec2\Ec2Client;
```

Kode Sampel

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

5. Tambahkan profil instance ke instans Amazon EC2 yang sedang berjalan atau dihentikan. Gunakan nama profil instance peran IAM Anda.

   Kode Sampel

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

Untuk informasi selengkapnya, lihat Peran IAM untuk Amazon EC2 di Panduan EC2 Pengguna Amazon.

Menggunakan peran IAM untuk tugas Amazon ECS

Tugas di Amazon Elastic Container Service (Amazon ECS) dapat mengambil peran IAM untuk melakukan panggilan API. AWS Ini adalah strategi untuk mengelola kredensi untuk aplikasi Anda untuk digunakan, mirip dengan cara profil EC2 instans Amazon memberikan kredensi ke instans Amazon. EC2

Alih-alih membuat dan mendistribusikan AWS kredensi jangka panjang ke container atau menggunakan peran EC2 instans Amazon, Anda dapat mengaitkan peran IAM yang menggunakan kredensi sementara dengan definisi tugas ECS atau operasi API. `RunTask`

Untuk informasi selengkapnya tentang penggunaan peran IAM yang dapat diasumsikan oleh tugas container, lihat topik peran IAM Tugas di Panduan Pengembang Amazon ECS. Untuk contoh penggunaan peran IAM tugas dalam bentuk definisi tugas `taskRoleArn` dalam, lihat Contoh definisi tugas juga di Panduan Pengembang Amazon ECS.

Dengan asumsi peran IAM di yang lain Akun AWS

Ketika Anda bekerja di Akun AWS (Akun A) dan ingin mengambil peran di akun lain (Akun B), Anda harus terlebih dahulu membuat peran IAM di Akun B. Peran ini memungkinkan entitas di akun Anda (Akun A) untuk melakukan tindakan tertentu di Akun B. Untuk informasi selengkapnya tentang akses lintas akun, lihat Tutorial: Mendelegasikan Akses di Seluruh AWS Akun Menggunakan Peran IAM.

Setelah Anda membuat peran di Akun B, catat ARN Peran. Anda akan menggunakan ARN ini ketika Anda mengambil peran dari Akun A. Anda mengambil peran menggunakan AWS kredenal yang terkait dengan entitas Anda di Akun A.

Buat AWS STS klien dengan kredensi untuk Anda. Akun AWS Berikut ini, kami menggunakan profil kredensial, tetapi Anda dapat menggunakan metode apa pun. Dengan AWS STS klien yang baru dibuat, panggil assume-role dan berikan SessionName kustom. Ambil kredensi sementara baru dari hasilnya. Kredenal bertahan satu jam secara default.

Kode Sampel

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

Untuk informasi selengkapnya, lihat Menggunakan Peran IAM atau AssumeRoledi Referensi AWS SDK for PHP API.

Menggunakan peran IAM dengan identitas web

Federasi Identitas Web memungkinkan pelanggan untuk menggunakan penyedia identitas pihak ketiga untuk otentikasi saat mengakses sumber daya. AWS Sebelum Anda dapat mengambil peran dengan identitas web, Anda harus membuat peran IAM dan mengonfigurasi penyedia identitas web (iDP). Untuk informasi selengkapnya, lihat Membuat Peran untuk Identitas Web atau Federasi OpenID Connect (Konsol).

Setelah membuat penyedia identitas dan membuat peran untuk identitas web Anda, gunakan AWS STS klien untuk mengautentikasi pengguna. Berikan webIdentityToken dan ProviderId untuk identitas Anda, dan ARN Peran untuk peran IAM dengan izin untuk pengguna.

Kode Sampel

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

Untuk informasi selengkapnya, lihat AssumeRoleWithWebIdentity—Federasi Melalui Penyedia Identitas Berbasis Web atau AssumeRoleWithWebIdentitydi Referensi AWS SDK for PHP API.

Asumsikan peran dengan profil

Tentukan profil di **`~/.aws/credentials`**

Anda dapat mengonfigurasi AWS SDK for PHP untuk menggunakan peran IAM dengan mendefinisikan profil di. `~/.aws/credentials`

Buat profil baru dengan `role_arn` pengaturan untuk peran yang ingin Anda asumsikan. Sertakan juga `source_profile` pengaturan untuk profil lain dengan kredensional yang memiliki izin untuk mengambil peran IAM. Untuk detail selengkapnya tentang setelan konfigurasi ini, lihat Mengasumsikan kredensi peran di Panduan Referensi Alat AWS SDKs dan Alat.

Misalnya, dalam hal berikut`~/.aws/credentials`, `project1` profil menetapkan `role_arn` dan menentukan `default` profil sebagai sumber kredensi untuk memverifikasi bahwa entitas yang terkait dengannya dapat mengambil peran tersebut.

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

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

Jika Anda menyetel variabel `AWS_PROFILE` lingkungan, atau Anda menggunakan `profile` parameter saat membuat instance klien layanan, peran yang ditentukan akan `project1` diasumsikan, menggunakan `default` profil sebagai kredenal sumber.

Cuplikan berikut menunjukkan penggunaan `profile` parameter dalam konstruktor. `S3Client` `S3Client`Akan memiliki izin yang terkait dengan peran yang terkait dengan `project1` profil.

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

Tentukan profil di **`~/.aws/config`**

`~/.aws/config`File ini juga dapat berisi profil yang ingin Anda asumsikan. Jika Anda mengatur variabel lingkunganAWS_SDK_LOAD_NONDEFAULT_CONFIG, SDK for PHP memuat profil dari file`config`. Bila AWS_SDK_LOAD_NONDEFAULT_CONFIG disetel, SDK memuat profil dari keduanya `~/.aws/config` dan`~/.aws/credentials`. Profil dari `~/.aws/credentials` dimuat terakhir dan diutamakan daripada profil dari `~/.aws/config` dengan nama yang sama. Profil dari salah satu lokasi dapat berfungsi sebagai `source_profile` atau profil yang akan diasumsikan.

Contoh berikut menggunakan `project1` profil yang ditentukan dalam `config` file dan `default` profil dalam `credentials` file. AWS_SDK_LOAD_NONDEFAULT_CONFIGIni juga diatur.

```
# Profile in ~/.aws/config.

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

```
# Profile in ~/.aws/credentials.

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

Saat `S3Client` konstruktor berjalan yang ditampilkan cuplikan berikut, peran yang ditentukan dalam `project1` profil akan diasumsikan menggunakan kredenal yang terkait dengan profil. `default`

```
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-1',
```

```
    'version' => '2006-03-01',
    'profile' => 'project1'
]);
```

Menggunakan kredensial sementara dari AWS STS SDK for PHP

AWS Security Token Service (AWS STS) memungkinkan Anda untuk meminta hak istimewa terbatas, kredensi sementara untuk pengguna IAM, atau untuk pengguna yang Anda autentikasi melalui federasi identitas. Untuk pemahaman yang lebih dalam, lihat Kredensial Keamanan Sementara di Panduan Pengguna IAM. Anda dapat menggunakan kredensi keamanan sementara untuk mengakses sebagian besar AWS layanan. Untuk daftar layanan yang menerima kredenal keamanan sementara, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Salah satu kasus penggunaan umum untuk kredensi sementara adalah memberikan akses aplikasi seluler atau sisi klien ke AWS sumber daya dengan mengautentikasi pengguna melalui penyedia identitas pihak ketiga (lihat Federasi Identitas Web).

Mendapatkan kredensi sementara

AWS STS memiliki beberapa operasi yang mengembalikan kredensi sementara, tetapi `GetSessionToken` operasi ini adalah yang paling sederhana untuk ditunjukkan. Cuplikan berikut mengambil kredensi sementara dengan memanggil `getSessionToken` metode klien STS PHP SDK.

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

$stsClient = $sdk->createSts();

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

Hasil untuk `GetSessionToken` dan AWS STS operasi lainnya selalu mengandung `'Credentials'` nilai. Jika Anda mencetak `$result` (misalnya dengan menggunakan`print_r($result)`), terlihat seperti berikut ini.

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

Memberikan kredensi sementara ke AWS SDK for PHP

Anda dapat menggunakan kredensi sementara dengan AWS klien lain dengan membuat instance klien dan meneruskan nilai yang diterima secara langsung. AWS STS

```
use Aws\S3\S3Client;

$result = $stsClient->getSessionToken();

$s3Client = new S3Client([
    'version'     => '2006-03-01',
    'region'      => 'us-west-2',
    'credentials' => [
        'key'    => $result['Credentials']['AccessKeyId'],
        'secret' => $result['Credentials']['SecretAccessKey'],
        'token'  => $result['Credentials']['SessionToken']
    ]
]);
```

Anda juga dapat membuat `Aws\Credentials\Credentials` objek dan menggunakannya saat membuat instance klien.

```
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
]);
```

Namun, cara terbaik untuk memberikan kredensi sementara adalah dengan menggunakan metode `createCredentials()` pembantu yang disertakan dengan. `StsClient` Metode ini mengekstrak data dari AWS STS hasil dan menciptakan `Credentials` objek untuk Anda.

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

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

Untuk informasi selengkapnya tentang alasan Anda mungkin perlu menggunakan kredensial sementara dalam aplikasi atau proyek, lihat Skenario untuk Memberikan Akses Sementara dalam dokumentasi. AWS STS

Buat klien anonim di SDK for PHP

Dalam beberapa kasus, Anda mungkin ingin membuat klien yang tidak terkait dengan kredensi apa pun. Ini memungkinkan Anda untuk membuat permintaan anonim ke layanan.

Misalnya, Anda dapat mengonfigurasi objek Amazon S3 dan CloudSearch domain Amazon untuk memungkinkan akses anonim.

Untuk membuat klien anonim, Anda mengatur `'credentials'` opsi `kefalse`.

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

// Makes an anonymous request. The object would need to be publicly
// readable for this to succeed.
```

```
$result = $s3Client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
]);
```

Gunakan ekstensi AWS Common Runtime (AWS CRT) di SDK for PHP

Pustaka AWS CRT menyediakan fungsionalitas dasar dengan kinerja yang baik dan jejak minimal untuk beberapa. AWS SDKs Topik ini membahas kapan AWS CRT digunakan oleh SDK for PHP dan cara menginstal ekstensi CRT. AWS

Saat Anda membutuhkan ekstensi AWS CRT diinstal

SDK for PHP menggunakan fungsi otorisasi dan checksum dari AWS pustaka CRT. Ekstensi AWS CRT diperlukan saat Anda bekerja dengan:

- Titik Akses Multi-Wilayah Amazon S3
- Titik akhir EventBridge global Amazon
- Algoritma checksum CRC-32C di Amazon Simple Storage Service (Amazon S3)

Jika Anda menggunakan fitur yang tercantum di atas dan ekstensi AWS CRT tidak diinstal di lingkungan PHP Anda, SDK for PHP akan melaporkan pesan kesalahan dan mengingatkan Anda untuk menginstal ekstensi.

Instal ekstensi AWS Common Runtime (AWS CRT)

Petunjuk tentang cara menginstal ekstensi AWS CRT tersedia di halaman utama GitHubrepositori untuk file. aws-crt-php

Menggunakan AWS SDK for PHP Versi 3

Bab ini memberikan panduan komprehensif tentang penggunaan AWS SDK for PHP Versi 3 untuk berinteraksi Layanan AWS secara efektif. Anda akan mempelajari teknik-teknik penting untuk membuat permintaan layanan, menangani kesalahan, dan memanfaatkan fitur SDK tingkat lanjut untuk membangun aplikasi PHP yang kuat. AWS

Apakah Anda sedang membangun aplikasi sederhana atau sistem yang kompleks, teknik yang dijelaskan dalam Bab ini akan membantu Anda mengoptimalkan kode Anda, meningkatkan penanganan kesalahan, dan menerapkan pola yang efisien untuk bekerja dengan Layanan AWS melalui PHP SDK.

Topik

- Membuat Layanan AWS permintaan menggunakan AWS SDK for PHP Versi 3
- Pemrograman asinkron menggunakan Versi 3 AWS SDK for PHP
- Menangani kesalahan dalam AWS SDK for PHP Versi 3
- Handler dan middleware di Versi 3 AWS SDK for PHP
- Streaming di AWS SDK for PHP Versi 3
- Menggunakan hasil paginasi di AWS SDK for PHP Versi 3
- Pelayan di Versi 3 AWS SDK for PHP
- JMESPath ekspresi dalam AWS SDK for PHP Versi 3

Membuat Layanan AWS permintaan menggunakan AWS SDK for PHP Versi 3

Ikhtisar alur kerja permintaan SDK

Bekerja dengan AWS SDK for PHP Versi 3 mengikuti pola yang konsisten di semua Layanan AWS. Alur kerja dasar melibatkan tiga langkah utama:

1. Buat klien layanan —Instantiate objek Klien untuk yang ingin Layanan AWS Anda gunakan.
2. Jalankan operasi —Metode panggilan pada klien yang sesuai dengan operasi di API layanan.
3. Hasil proses —Bekerja dengan objek Result seperti array yang dikembalikan pada kesuksesan, atau tangani Pengecualian apa pun yang dilemparkan pada kegagalan.

Bagian berikut menjelaskan masing-masing langkah ini secara rinci, dimulai dengan cara membuat dan mengkonfigurasi klien layanan.

Membuat klien layanan dasar

Anda dapat membuat klien dengan meneruskan array pilihan asosiatif ke konstruktor klien.

Impor

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

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

Kode Sampel

```
//Create an S3Client
$s3 = new Aws\S3\S3Client([
    'region' => 'us-east-2'  // Since version 3.277.10 of the SDK,
]);                          // the 'version' parameter defaults to 'latest'.
```

Informasi tentang parameter “versi” opsional tersedia di topik opsi konfigurasi.

Perhatikan bahwa kami tidak secara eksplisit memberikan kredensil kepada klien. Itu karena SDK menggunakan rantai penyedia kredensi default untuk mencari informasi kredensi.

Semua opsi konfigurasi klien umum dijelaskan secara rinci diOpsi konstruktor klien untuk AWS SDK for PHP Versi 3. Array opsi yang diberikan kepada klien dapat bervariasi berdasarkan klien mana yang Anda buat. Opsi konfigurasi klien kustom ini dijelaskan dalam dokumentasi API untuk setiap klien.

Sementara contoh di atas menunjukkan pembuatan klien dasar, Anda dapat menyesuaikan klien layanan Anda untuk memenuhi persyaratan tertentu. Untuk informasi lebih rinci tentang mengkonfigurasi klien layanan melalui kode, lihatMengkonfigurasi klien layanan dalam kode untuk AWS SDK for PHP Versi 3. Jika Anda perlu mengkonfigurasi klien layanan menggunakan file konfigurasi eksternal atau variabel lingkungan, lihatMengkonfigurasi klien layanan untuk AWS SDK for PHP Versi 3 secara eksternal.

Membuat permintaan

Anda dapat membuat permintaan layanan dengan memanggil metode dengan nama yang sama pada objek klien. Misalnya, untuk melakukan PutObjectoperasi Amazon S3, Anda memanggil metode. `Aws\S3\S3Client::putObject()`

Impor

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

use Aws\S3\S3Client;
```

Kode Sampel

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2'
];

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

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

// Send a PutObject request and get the result object.
$result = $s3Client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'my-key',
    'Body' => 'this is the body!'
]);

// Download the contents of the object.
$result = $s3Client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'my-key'
]);

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

Operasi yang tersedia untuk klien dan struktur input dan output didefinisikan pada runtime berdasarkan file deskripsi layanan. Saat membuat klien, jika Anda tidak memberikan `version` parameter (misalnya., “2006-03-01" atau “terbaru”) dari model layanan, klien default ke versi terbaru. SDK menemukan file konfigurasi yang sesuai berdasarkan versi yang disediakan.

Metode operasi seperti `putObject()` semua menerima argumen tunggal, array asosiatif yang mewakili parameter operasi. Struktur array ini (dan struktur objek hasil) didefinisikan untuk setiap operasi dalam Dokumentasi API SDK (misalnya, lihat dokumen API untuk operasi putObject).

Opsi penangan HTTP

Anda juga dapat menyempurnakan bagaimana penangan HTTP yang mendasari mengeksekusi permintaan dengan menggunakan parameter khusus. `@http` Opsi yang dapat Anda sertakan dalam `@http` parameter sama dengan yang dapat Anda atur saat Anda membuat instance klien dengan opsi klien “http”.

```
// Send the request through a proxy
$result = $s3Client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'my-key',
    'Body'   => 'this is the body!',
    '@http'  => [
        'proxy' => 'http://192.168.16.1:10'
    ]
]);
```

Bekerja dengan objek hasil

Mengeksekusi operasi yang berhasil mengembalikan `Aws\Result` objek. Alih-alih mengembalikan data XHTML atau JSON mentah dari suatu layanan, SDK memaksa data respons ke dalam struktur array asosiatif. Ini menormalkan beberapa aspek data berdasarkan pengetahuannya tentang layanan spesifik dan struktur respons yang mendasarinya.

Anda dapat mengakses data dari `AWS\Result` objek seperti array PHP asosiatif.

Impor

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

Kode Sampel

```
// Use the us-east-2 region and latest version of each client.
$sharedConfig = [
    'profile' => 'default',
    'region' => 'us-east-2',
];

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

// Use an Aws\Sdk class to create the S3Client object.
$s3 = $sdk->createS3();
$result = $s3->listBuckets();
foreach ($result['Buckets'] as $bucket) {
    echo $bucket['Name'] . "\n";
}

// Convert the result object to a PHP array
$array = $result->toArray();
```

Isi objek hasil tergantung pada operasi yang dijalankan dan versi layanan. Struktur hasil dari setiap operasi API didokumentasikan dalam dokumen API untuk setiap operasi.

SDK terintegrasi dengan JMESPath, DSL yang digunakan untuk mencari dan memanipulasi data JSON atau, dalam kasus kami, array PHP. Objek hasil berisi `search()` metode yang dapat Anda gunakan untuk mengekstrak data secara lebih deklaratif dari hasilnya.

Kode Sampel

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

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

Objek perintah dalam AWS SDK for PHP Versi 3

AWS SDK for PHP Menggunakan pola perintah untuk merangkum parameter dan handler yang akan digunakan untuk mentransfer permintaan HTTP di lain waktu.

Penggunaan perintah secara implisit

Jika Anda memeriksa kelas klien apa pun, Anda dapat melihat bahwa metode yang sesuai dengan operasi API sebenarnya tidak ada. Mereka diimplementasikan menggunakan metode `__call()` sihir. Metode semu ini sebenarnya adalah pintasan yang merangkum penggunaan objek perintah SDK.

Anda biasanya tidak perlu berinteraksi dengan objek perintah secara langsung. Saat Anda memanggil metode seperti`Aws\S3\S3Client::putObject()`, SDK sebenarnya membuat `Aws\CommandInterface` objek berdasarkan parameter yang disediakan, mengeksekusi perintah, dan mengembalikan `Aws\ResultInterface` objek yang diisi (atau melempar pengecualian pada kesalahan). Alur serupa terjadi ketika memanggil salah satu `Async` metode klien (misalnya,`Aws\S3\S3Client::putObjectAsync()`): klien membuat perintah berdasarkan parameter yang disediakan, membuat serial permintaan HTTP, memulai permintaan, dan mengembalikan janji.

Contoh-contoh berikut secara fungsional setara.

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

$params = [
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key'    => 'baz',
    'Body'   => 'bar'
];

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

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

Parameter perintah

Semua perintah mendukung beberapa parameter khusus yang bukan bagian dari API layanan tetapi mengontrol perilaku SDK.

`@http`

Dengan menggunakan parameter ini, dimungkinkan untuk menyempurnakan bagaimana penangan HTTP yang mendasari mengeksekusi permintaan. Opsi yang dapat Anda sertakan dalam `@http` parameter sama dengan yang dapat Anda atur saat Anda membuat instance klien dengan opsi klien "http".

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

`@retries`

Seperti opsi klien "coba ulang", `@retries` mengontrol berapa kali perintah dapat dicoba ulang sebelum dianggap gagal. Setel ke `0` untuk menonaktifkan percobaan ulang.

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

> **Note**
>
> Jika Anda telah menonaktifkan percobaan ulang pada klien, Anda tidak dapat mengaktifkannya secara selektif pada perintah individual yang diteruskan ke klien tersebut.

Membuat objek perintah

Anda dapat membuat perintah menggunakan `getCommand()` metode klien. Itu tidak segera mengeksekusi atau mentransfer permintaan HTTP, tetapi hanya dieksekusi ketika diteruskan ke `execute()` metode klien. Ini memberi Anda kesempatan untuk memodifikasi objek perintah sebelum menjalankan perintah.

```
$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);
```

Perintah **HandlerList**

Ketika perintah dibuat dari klien, itu diberikan klon dari `Aws\HandlerList` objek klien. Perintah diberikan klon dari daftar handler klien untuk memungkinkan perintah menggunakan middleware kustom dan handler yang tidak mempengaruhi perintah lain yang dijalankan klien.

Ini berarti Anda dapat menggunakan klien HTTP yang berbeda per perintah (misalnya,`Aws\MockHandler`) dan menambahkan perilaku khusus per perintah melalui middleware. Contoh berikut menggunakan a `MockHandler` untuk membuat hasil tiruan alih-alih mengirim permintaan HTTP yang sebenarnya.

```
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'
```

Selain mengubah handler yang digunakan perintah, Anda juga dapat menyuntikkan middleware khusus ke perintah. Contoh berikut menggunakan `tap` middleware, yang berfungsi sebagai pengamat dalam daftar handler.

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

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

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

// Append the middleware to the "sign" step of the handler list. The sign
// step is the last step before transferring an HTTP request.
$list->append('sign', $middleware);

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

`CommandPool`

`Aws\CommandPool`Ini memungkinkan Anda untuk menjalankan perintah secara bersamaan menggunakan iterator yang menghasilkan `Aws\CommandInterface` objek. `CommandPool`Ini memastikan bahwa sejumlah perintah yang konstan dijalankan secara bersamaan saat mengulang perintah di kolam (saat perintah selesai, lebih banyak dijalankan untuk memastikan ukuran kolam konstan).

Berikut adalah contoh yang sangat sederhana dari hanya mengirim beberapa perintah menggunakan`CommandPool`.

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

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

$bucket = 'amzn-s3-demo-bucket';
$commands = [
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'a']),
```

```
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'b']),
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'c'])
];

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

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

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

Contoh itu cukup kurang bertenaga untuk. `CommandPool` Mari kita coba contoh yang lebih kompleks. Katakanlah Anda ingin mengunggah file pada disk ke bucket Amazon S3. Untuk mendapatkan daftar file dari disk, kita bisa menggunakan PHP`DirectoryIterator`. Iterator ini menghasilkan objek`SplFileInfo`. `CommandPool`Menerima iterator yang menghasilkan `Aws\CommandInterface` objek, jadi kami memetakan `SplFileInfo` objek untuk mengembalikan objek. `Aws\CommandInterface`

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

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

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

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

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

// Create a generator that converts the SplFileInfo objects into
```

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

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

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

```
        echo "Failed {$iterKey}: {$reason}\n";
    },
]);

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

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

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

`CommandPool`konfigurasi

`Aws\CommandPool`Konstruktor menerima berbagai opsi konfigurasi.

konkurensi (dapat dipanggil | int)

Jumlah maksimum perintah untuk mengeksekusi secara bersamaan. Menyediakan fungsi untuk mengubah ukuran kolam secara dinamis. Fungsi ini disediakan jumlah permintaan tertunda saat ini dan diharapkan untuk mengembalikan bilangan bulat yang mewakili batas ukuran kolam baru.

sebelum (dapat dipanggil)

Fungsi untuk memanggil sebelum mengirim setiap perintah. `before`Fungsi menerima perintah dan kunci iterator perintah. Anda dapat mengubah perintah sesuai kebutuhan dalam `before` fungsi sebelum mengirim perintah.

terpenuhi (dapat dipanggil)

Berfungsi untuk memanggil ketika janji terpenuhi. Fungsi ini disediakan objek hasil, ID iterator yang hasilnya berasal, dan janji agregat yang dapat diselesaikan atau ditolak jika Anda perlu melakukan hubungan pendek pada pool.

ditolak (dapat dipanggil)

Berfungsi untuk memanggil ketika janji ditolak. Fungsi ini disediakan `Aws\Exception` objek, ID iterator tempat pengecualian berasal, dan janji agregat yang dapat diselesaikan atau ditolak jika Anda perlu melakukan hubungan pendek pada kumpulan.

Pengumpulan sampah manual antar perintah

Jika Anda mencapai batas memori dengan kumpulan perintah besar, ini mungkin karena referensi siklik yang dihasilkan oleh SDK belum dikumpulkan oleh pengumpul sampah PHP ketika batas memori Anda tercapai. Memanggil algoritma pengumpulan secara manual di antara perintah memungkinkan siklus dikumpulkan sebelum mencapai batas itu. Contoh berikut membuat sebuah `CommandPool` yang memanggil algoritma koleksi menggunakan callback sebelum mengirim setiap perintah. Perhatikan bahwa memanggil pengumpul sampah memang datang dengan biaya kinerja, dan penggunaan optimal akan tergantung pada kasus penggunaan dan lingkungan Anda.

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

Pemrograman asinkron menggunakan Versi 3 AWS SDK for PHP

Anda dapat mengirim perintah secara bersamaan menggunakan fitur asinkron SDK. Anda dapat mengirim permintaan secara asinkron dengan akhiran nama operasi dengan. `Async` Ini memulai permintaan dan mengembalikan janji.

Janji dipenuhi dengan objek hasil pada keberhasilan atau ditolak dengan pengecualian pada kegagalan. Ini memungkinkan Anda untuk membuat beberapa janji dan meminta mereka mengirim permintaan HTTP secara bersamaan ketika penangan HTTP yang mendasari mentransfer permintaan.

Impor

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

Kode Sampel

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
```

```
]);
// Use an Aws\Sdk class to create the S3Client object.
$s3Client = $sdk->createS3();
//Listing all S3 Bucket
$CompleteSynchronously = $s3Client->listBucketsAsync();
// Block until the result is ready.
$CompleteSynchronously = $CompleteSynchronously->wait();
```

Anda dapat memaksakan janji untuk menyelesaikan secara serempak dengan menggunakan `wait` metode janji. Memaksa janji untuk menyelesaikan juga "membuka" status janji secara default, yang berarti itu akan mengembalikan hasil janji atau membuang pengecualian yang ditemui. Saat memanggil `wait()` janji, proses memblokir sampai permintaan HTTP selesai dan hasilnya diisi atau pengecualian dilemparkan.

Saat menggunakan SDK dengan pustaka loop peristiwa, jangan blokir hasil. Sebagai gantinya, gunakan `then()` metode hasil untuk mengakses janji yang diselesaikan atau ditolak saat operasi selesai.

Impor

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

Kode Sampel

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

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

```
    });
```

Janji dalam AWS SDK for PHP Versi 3

AWS SDK for PHP Penggunaan menjanjikan untuk memungkinkan alur kerja asinkron, dan asinkron ini memungkinkan permintaan HTTP dikirim secara bersamaan. Spesifikasi janji yang digunakan oleh SDK adalah Promises/A +.

Apa itu janji?

Janji merupakan hasil akhir dari operasi asinkron. Cara utama berinteraksi dengan janji adalah melalui `then` metodenya. Metode ini mendaftarkan callback untuk menerima nilai akhir janji atau alasan mengapa janji tidak dapat dipenuhi.

AWS SDK for PHP Bergantung pada paket Guzzlehttp/promises Composer untuk implementasi janjinya. Guzzle menjanjikan mendukung alur kerja pemblokiran dan non-pemblokiran dan dapat digunakan dengan loop peristiwa non-pemblokiran apa pun.

> **Note**
>
> Permintaan HTTP dikirim secara bersamaan dalam AWS SDK for PHP menggunakan satu utas, di mana panggilan non-pemblokiran digunakan untuk mentransfer satu atau beberapa permintaan HTTP saat bereaksi terhadap perubahan status (misalnya, memenuhi atau menolak janji).

Janji di SDK

Janji digunakan di seluruh SDK. Misalnya, janji digunakan di sebagian besar abstraksi tingkat tinggi yang disediakan oleh SDK: paginator, pelayan, kumpulan perintah, unggahan multipart, transfer S3, dan sebagainya. directory/bucket

Semua klien yang SDK memberikan janji pengembalian saat Anda memanggil salah satu metode `Async` akhiran. Misalnya, kode berikut menunjukkan cara membuat janji untuk mendapatkan hasil operasi Amazon `DescribeTable` DynamoDB.

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

```
]);

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

Perhatikan bahwa Anda dapat menelepon salah satu `describeTable` atau`describeTableAsync`. Metode ini adalah `__call` metode ajaib pada klien yang didukung oleh model API dan `version` nomor yang terkait dengan klien. Dengan memanggil metode seperti `describeTable` tanpa `Async` akhiran, klien akan memblokir saat mengirim permintaan HTTP dan mengembalikan `Aws\ResultInterface` objek atau melempar file. `Aws\Exception\AwsException` Dengan akhiran nama operasi dengan `Async` (yaitu,`describeTableAsync`) klien akan membuat janji yang akhirnya dipenuhi dengan `Aws\ResultInterface` objek atau ditolak dengan. `Aws\Exception\AwsException`

> **Important**
>
> Ketika janji dikembalikan, hasilnya mungkin sudah tiba (misalnya, saat menggunakan penangan tiruan), atau permintaan HTTP mungkin belum dimulai.

Anda dapat mendaftarkan callback dengan janji dengan menggunakan `then` metode ini. Metode ini menerima dua callback`$onRejected`, `$onFulfilled` dan keduanya bersifat opsional. `$onFulfilled`Panggilan balik dipanggil jika janji terpenuhi, dan `$onRejected` panggilan balik dipanggil jika janji ditolak (artinya gagal).

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

Menjalankan perintah secara bersamaan

Beberapa janji dapat disusun bersama sedemikian rupa sehingga dieksekusi secara bersamaan. Ini dapat dicapai dengan mengintegrasikan SDK dengan loop peristiwa non-pemblokiran, atau dengan membangun beberapa janji dan menunggu mereka selesai secara bersamaan.

```
use GuzzleHttp\Promise\Utils;

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

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

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

// Wait for both promises to complete.
$results = Utils::unwrap($promises);

// Notice that this method will maintain the input array keys.
var_dump($results['buckets']->toArray());
var_dump($results['tables']->toArray());
```

> **Note**
>
> CommandPoolIni menyediakan mekanisme yang lebih kuat untuk mengeksekusi beberapa operasi API secara bersamaan.

Janji rantai

Salah satu aspek terbaik dari janji adalah bahwa mereka dapat dikomposisi, memungkinkan Anda untuk membuat pipeline transformasi. Janji disusun dengan merantai `then` callback dengan callback berikutnya`then`. Nilai pengembalian `then` metode adalah janji yang dipenuhi atau ditolak berdasarkan hasil callback yang diberikan.

```
$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**
>
> Nilai pengembalian callback janji adalah `$value` argumen yang diberikan ke janji hilir. Jika Anda ingin memberikan nilai ke rantai janji hilir, Anda harus mengembalikan nilai dalam fungsi callback.

Penerusan penolakan

Anda dapat mendaftarkan callback untuk memanggil saat janji ditolak. Jika pengecualian dilemparkan dalam panggilan balik apa pun, janji ditolak dengan pengecualian dan janji berikutnya dalam rantai ditolak dengan pengecualian. Jika Anda berhasil mengembalikan nilai dari `$onRejected` callback, janji berikutnya dalam rantai janji dipenuhi dengan nilai pengembalian dari `$onRejected` callback.

Menunggu janji

Anda dapat secara sinkron memaksa janji untuk diselesaikan dengan menggunakan `wait` metode janji.

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

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

Jika pengecualian ditemukan saat menjalankan `wait` fungsi janji, janji ditolak dengan pengecualian dan pengecualian dilemparkan.

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

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

`wait`Memanggil janji yang telah dipenuhi tidak memicu fungsi tunggu. Ini hanya mengembalikan nilai yang dikirimkan sebelumnya.

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

`wait`Memanggil janji yang telah ditolak memberikan pengecualian. Jika alasan penolakan adalah contoh dari `\Exception` alasan dilemparkan. Jika tidak, a `GuzzleHttp\Promise\RejectionException` dilemparkan dan alasannya dapat diperoleh dengan memanggil `getReason` metode pengecualian.

> **Note**
>
> Panggilan operasi API di AWS SDK for PHP ditolak dengan subclass `Aws\Exception\AwsException` kelas. Namun, ada kemungkinan alasan yang dikirim ke suatu `then` metode berbeda karena penambahan middleware khusus yang mengubah alasan penolakan.

Membatalkan janji

Janji dapat dibatalkan menggunakan `cancel()` metode janji. Jika janji telah diselesaikan, panggilan tidak `cancel()` akan berpengaruh. Membatalkan janji membatalkan janji dan janji apa pun yang menunggu pengiriman dari janji. Janji yang dibatalkan ditolak dengan a. `GuzzleHttp\Promise\RejectionException`

Menggabungkan janji

Anda dapat menggabungkan janji menjadi janji agregat untuk membangun alur kerja yang lebih canggih. `guzzlehttp/promise`Paket berisi berbagai fungsi yang dapat Anda gunakan untuk menggabungkan janji.

Anda dapat menemukan dokumentasi API untuk semua fungsi pengumpulan janji di namespace-GuzzleHttp .Promise.

masing-masing dan each_limit

Gunakan CommandPoolketika Anda memiliki antrian tugas `Aws\CommandInterface` perintah untuk melakukan bersamaan dengan ukuran kolam tetap (perintah dapat berada di memori atau dihasilkan oleh iterator malas). `CommandPool`Ini memastikan bahwa sejumlah perintah tetap dikirim secara bersamaan sampai iterator yang disediakan habis.

`CommandPool`Bekerja hanya dengan perintah yang dijalankan oleh klien yang sama. Anda dapat menggunakan `GuzzleHttp\Promise\each_limit` fungsi untuk melakukan perintah kirim dari klien yang berbeda secara bersamaan menggunakan ukuran kolam tetap.

```
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();
```

Coroutine janji

Salah satu fitur yang lebih kuat dari pustaka janji Guzzle adalah memungkinkan Anda menggunakan coroutine janji yang membuat penulisan alur kerja asinkron tampak lebih seperti menulis alur kerja sinkron tradisional. Bahkan, AWS SDK for PHP penggunaan coroutine menjanjikan di sebagian besar abstraksi tingkat tinggi.

Bayangkan Anda ingin membuat beberapa bucket dan mengunggah file ke bucket saat bucket tersedia, dan Anda ingin melakukan ini semua secara bersamaan sehingga itu terjadi secepat mungkin. Anda dapat melakukan ini dengan mudah dengan menggabungkan beberapa janji coroutine bersama-sama menggunakan fungsi `all()` promise.

```
use GuzzleHttp\Promise;

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

// Create the following buckets
$buckets = ['amzn-s3-demo-bucket1', 'amzn-s3-demo-bucket2', 'amzn-s3-demo-bucket3'];
$promises = [];

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

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

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

Menangani kesalahan dalam AWS SDK for PHP Versi 3

Penanganan Kesalahan Sinkron

Jika terjadi kesalahan saat melakukan operasi, pengecualian dilemparkan. Untuk alasan ini, jika Anda perlu menangani kesalahan dalam kode Anda, gunakan`try`/`catch`blok di sekitar operasi Anda. SDK melempar pengecualian khusus layanan saat terjadi kesalahan.

Contoh berikut menggunakan`Aws\S3\S3Client`. Jika ada kesalahan, pengecualian yang dilemparkan akan menjadi tipe`Aws\S3\Exception\S3Exception`. Semua pengecualian khusus layanan yang dilemparkan SDK diperluas dari kelas. `Aws\Exception\AwsException` Kelas ini berisi informasi yang berguna tentang kegagalan, termasuk request-id, kode kesalahan, dan jenis kesalahan. Catatan untuk beberapa layanan yang mendukungnya, data respons dipaksa menjadi struktur array asosiatif (mirip dengan `Aws\Result` objek), yang dapat diakses seperti array asosiatif PHP normal. `toArray()`Metode ini akan mengembalikan data tersebut, jika ada.

Impor

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

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

Kode Sampel

```
// Create an SDK class used to share configuration across clients.
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2'
]);

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

```
try {
    $s3Client->createBucket(['Bucket' => 'amzn-s3-demo-bucket']);
} catch (S3Exception $e) {
    // Catch an S3 specific exception.
    echo $e->getMessage();
} catch (AwsException $e) {
    // This catches the more generic AwsException. You can grab information
    // from the exception using methods of the exception object.
    echo $e->getAwsRequestId() . "\n";
    echo $e->getAwsErrorType() . "\n";
    echo $e->getAwsErrorCode() . "\n";

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

Penanganan kesalahan asinkron

Pengecualian tidak dilemparkan saat mengirim permintaan asinkron. Sebagai gantinya, Anda harus menggunakan `then()` atau `otherwise()` metode janji yang dikembalikan untuk menerima hasil atau kesalahan.

Impor

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

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

Kode Sampel

```
//Asynchronous Error Handling
$promise = $s3Client->createBucketAsync(['Bucket' => 'amzn-s3-demo-bucket']);
$promise->otherwise(function ($reason) {
    var_dump($reason);
});
```

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

Anda dapat “membuka” janji dan menyebabkan pengecualian dilemparkan sebagai gantinya.

Impor

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

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

Kode Sampel

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

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

Handler dan middleware di Versi 3 AWS SDK for PHP

Mekanisme utama untuk memperluas AWS SDK for PHP adalah melalui handler dan middleware. Setiap kelas klien SDK memiliki `Aws\HandlerList` instance yang dapat diakses melalui `getHandlerList()` metode klien. Anda dapat mengambil klien `HandlerList` dan memodifikasinya untuk menambah atau menghapus perilaku klien.

Penangan

Handler adalah fungsi yang melakukan transformasi aktual dari perintah dan permintaan menjadi hasil. Handler biasanya mengirimkan permintaan HTTP. Handler dapat disusun dengan

middleware untuk meningkatkan perilaku mereka. Handler adalah fungsi yang menerima `Aws\CommandInterface` dan a `Psr\Http\Message\RequestInterface` dan mengembalikan janji yang dipenuhi dengan `Aws\ResultInterface` atau ditolak dengan alasan. `Aws\Exception\AwsException`

Berikut adalah handler yang mengembalikan hasil tiruan yang sama untuk setiap panggilan.

```
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);
};
```

Anda kemudian dapat menggunakan handler ini dengan klien SDK dengan memberikan `handler` opsi di konstruktor klien.

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

Anda juga dapat mengubah handler klien setelah dibangun menggunakan `setHandler` metode. `Aws\ClientInterface`

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

> **Note**
>
> Untuk mengubah handler klien multi-wilayah setelah dibangun, gunakan `useCustomHandler` metode. `Aws\MultiRegionClient`
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

Pawang tiruan

Sebaiknya gunakan `MockHandler` saat menulis tes yang menggunakan SDK. Anda dapat menggunakan `Aws\MockHandler` untuk mengembalikan hasil yang diejek atau melempar pengecualian tiruan. Anda mengantrekan hasil atau pengecualian, dan MockHandler mengantrekannya dalam urutan 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

Middleware adalah jenis khusus dari fungsi tingkat tinggi yang menambah perilaku mentransfer perintah, dan mendelegasikan ke handler “berikutnya”. Fungsi middleware menerima `Aws\CommandInterface` dan `Psr\Http\Message\RequestInterface` dan mengembalikan janji

yang dipenuhi dengan `Aws\ResultInterface` atau ditolak dengan alasan. `Aws\Exception \AwsException`

Middleware adalah fungsi tingkat tinggi yang memodifikasi perintah, permintaan, atau hasil saat melewati middleware. Middleware mengambil bentuk berikut.

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

Middleware menerima perintah untuk mengeksekusi dan objek permintaan opsional. Middleware dapat memilih untuk menambah permintaan dan perintah atau membiarkannya apa adanya. Middleware kemudian memanggil pegangan berikutnya dalam rantai atau dapat memilih untuk melakukan hubungan pendek pada handler berikutnya dan mengembalikan janji. Janji yang dibuat dengan memanggil handler berikutnya kemudian dapat ditambah menggunakan `then` metode janji untuk memodifikasi hasil atau kesalahan akhirnya sebelum mengembalikan janji mencadangkan tumpukan middleware.

HandlerList

SDK menggunakan `Aws\HandlerList` untuk mengelola middleware dan handler yang digunakan saat menjalankan perintah. Setiap klien SDK memiliki`HandlerList`, dan ini `HandlerList` dikloning dan ditambahkan ke setiap perintah yang dibuat klien. Anda dapat melampirkan middleware dan handler default untuk digunakan untuk setiap perintah yang dibuat oleh klien dengan menambahkan middleware ke klien. `HandlerList` Anda dapat menambah dan menghapus middleware dari perintah tertentu dengan memodifikasi yang `HandlerList` dimiliki oleh perintah tertentu.

A **`HandlerList`** merupakan tumpukan middleware yang digunakan untuk membungkus handler. Untuk membantu mengelola daftar middleware dan urutan di mana mereka membungkus handler, `HandlerList` memecah tumpukan middleware menjadi langkah-langkah bernama yang mewakili bagian dari siklus hidup mentransfer perintah:

1. `init`- Tambahkan parameter default
2. `validate`- Validasi parameter yang diperlukan
3. `build`- Serialisasi permintaan HTTP untuk mengirim
4. `sign`- Tanda tangani permintaan HTTP serial
5. **<**handler**>**(bukan langkah, tetapi melakukan transfer yang sebenarnya)

inisialisasi

Langkah siklus hidup ini mewakili inisialisasi perintah, dan permintaan belum diserialisasi. Langkah ini biasanya digunakan untuk menambahkan parameter default ke perintah.

Anda dapat menambahkan middleware ke `init` langkah menggunakan `prependInit` metode `appendInit` dan, di mana `appendInit` menambahkan middleware ke akhir `prepend` daftar sambil `prependInit` menambahkan middleware ke bagian depan daftar. `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');
```

validasi

Langkah siklus hidup ini digunakan untuk memvalidasi parameter input dari sebuah perintah.

Anda dapat menambahkan middleware ke `validate` langkah menggunakan `prependValidate` metode `appendValidate` dan, di mana `appendValidate` menambahkan middleware ke akhir `validate` daftar sambil `prependValidate` menambahkan middleware ke bagian depan daftar. `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

Langkah siklus hidup ini digunakan untuk membuat serial permintaan HTTP untuk perintah yang sedang dijalankan. Acara siklus hidup hilir akan menerima perintah dan permintaan HTTP PSR-7.

Anda dapat menambahkan middleware ke `build` langkah menggunakan `prependBuild` metode `appendBuild` dan, di mana `appendBuild` menambahkan middleware ke akhir `build` daftar sambil `prependBuild` menambahkan middleware ke bagian depan daftar. `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');
```

tanda tangan

Langkah siklus hidup ini biasanya digunakan untuk menandatangani permintaan HTTP sebelum dikirim melalui kawat. Anda biasanya harus menahan diri dari mutasi permintaan HTTP setelah ditandatangani untuk menghindari kesalahan tanda tangan.

Ini adalah langkah terakhir `HandlerList` sebelum permintaan HTTP ditransfer oleh handler.

Anda dapat menambahkan middleware ke `sign` langkah menggunakan `prependSign` metode `appendSign` dan, di mana `appendSign` menambahkan middleware ke akhir `sign` daftar sambil `prependSign` menambahkan middleware ke bagian depan daftar. `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 yang tersedia

SDK menyediakan beberapa middleware yang dapat Anda gunakan untuk meningkatkan perilaku klien atau untuk mengamati eksekusi perintah.

mapCommand

`Aws\Middleware::mapCommand`Middleware berguna ketika Anda perlu memodifikasi perintah sebelum perintah diserialisasi sebagai permintaan HTTP. Misalnya, `mapCommand` dapat digunakan untuk melakukan validasi atau menambahkan parameter default. `mapCommand`Fungsi menerima callable yang menerima `Aws\CommandInterface` objek dan mengembalikan objek. `Aws\CommandInterface`

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

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

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

mapRequest

`Aws\Middleware::mapRequest`Middleware berguna ketika Anda perlu memodifikasi permintaan setelah serial tetapi sebelum dikirim. Misalnya, ini dapat digunakan untuk menambahkan header HTTP kustom ke permintaan. `mapRequest`Fungsi menerima callable yang menerima `Psr\Http\Message\RequestInterface` argumen dan mengembalikan objek. `Psr\Http\Message\RequestInterface`

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

// Create a command so that we can access the handler list
$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'amzn-s3-demo-bucket'
]);

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

Sekarang ketika perintah dijalankan, itu dikirim dengan header khusus.

> **Important**
>
> Perhatikan bahwa middleware ditambahkan ke daftar handler di akhir langkah. `build` Ini untuk memastikan bahwa permintaan telah dibuat sebelum middleware ini dipanggil.

mapResult

`Aws\Middleware::mapResult`Middleware berguna ketika Anda perlu memodifikasi hasil eksekusi perintah. `mapResult`Fungsi menerima callable yang menerima `Aws\ResultInterface` argumen dan mengembalikan objek. `Aws\ResultInterface`

```
use Aws\Middleware;
```

```
use Aws\ResultInterface;

$command = $s3Client->getCommand('HeadObject', [
    'Key'    => 'test',
    'Bucket' => 'amzn-s3-demo-bucket'
]);

$command->getHandlerList()->appendSign(
    Middleware::mapResult(function (ResultInterface $result) {
        // Add a custom value to the result
        $result['foo'] = 'bar';
        return $result;
    })
);
```

Sekarang ketika perintah dijalankan, hasil yang dikembalikan akan berisi `foo` atribut.

riwayat

`history`Middleware berguna untuk menguji bahwa SDK menjalankan perintah yang Anda harapkan, mengirim permintaan HTTP yang Anda harapkan, dan menerima hasil yang Anda harapkan. Ini pada dasarnya middleware yang bertindak mirip dengan sejarah browser 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));
```

Kontainer `Aws\History` riwayat menyimpan 10 entri secara default sebelum membersihkan entri. Anda dapat menyesuaikan jumlah entri dengan meneruskan jumlah entri untuk bertahan ke konstruktor.

```
// Create a history container that stores 20 entries
```

```
$history = new History(20);
```

Anda dapat memeriksa wadah riwayat setelah mengeksekusi permintaan yang melewati middleware riwayat.

```
// 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();
```

keran

`tap`Middleware digunakan sebagai pengamat. Anda dapat menggunakan middleware ini untuk menjalankan fungsi saat mengirim perintah melalui rantai middleware. `tap`Fungsi menerima

callable yang menerima `Aws\CommandInterface` dan opsional `Psr\Http\Message\RequestInterface` yang sedang dijalankan.

```
use Aws\Middleware;

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

$handlerList = $s3->getHandlerList();

// Create a tap middleware that observes the command at a specific step
$handlerList->appendInit(
    Middleware::tap(function (CommandInterface $cmd, RequestInterface $req = null) {
        echo 'About to send: ' . $cmd->getName() . "\n";
        if ($req) {
            echo 'HTTP method: ' . $request->getMethod() . "\n";
        }
    }
);
```

Membuat handler kustom

Handler hanyalah fungsi yang menerima objek dan `Aws\CommandInterface` `Psr\Http\Message\RequestInterface` objek, dan mengembalikan a `GuzzleHttp\Promise\PromiseInterface` yang dipenuhi dengan `Aws\ResultInterface` atau ditolak dengan. `Aws\Exception\AwsException`

Meskipun SDK memiliki beberapa `@http` opsi, penangan hanya perlu mengetahui cara menggunakan opsi berikut:

- connect_timeout
- debug
- decode_content (opsional)
- delay
- kemajuan (opsional)
- proksi
- wastafel

- sinkron (opsional)
- aliran (opsional)
- batas waktu
- verifikasi
- http_stats_receiver (opsional) - Fungsi untuk memanggil dengan array asosiatif statistik transfer HTTP jika diminta menggunakan parameter konfigurasi statistik.

Kecuali jika opsi ditentukan sebagai opsional, penangan HARUS dapat menangani opsi atau HARUS mengembalikan janji yang ditolak.

Selain menangani `@http` opsi tertentu, handler HARUS menambahkan `User-Agent` header yang mengambil formulir berikut, di mana “3.X” dapat diganti dengan `Aws\Sdk::VERSION` dan “HandlerSpecificData/version...” harus diganti dengan string User-Agent khusus penangan Anda.

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

Streaming di AWS SDK for PHP Versi 3

Sebagai bagian dari integrasi standar pesan HTTP PSR-7, PSR-7 AWS SDK for PHP menggunakan PSR-7 StreamInterface secara internal sebagai abstraksi atas aliran PHP. Setiap perintah dengan bidang input didefinisikan sebagai gumpalan, seperti `Body` parameter pada PutObject perintah S3::, dapat dipenuhi dengan string, sumber daya aliran PHP, atau instance dari. `Psr\Http\Message\StreamInterface`

> **Warning**
>
> SDK mengambil kepemilikan sumber daya aliran PHP mentah yang disediakan sebagai parameter input ke perintah. Aliran dikonsumsi dan ditutup atas nama Anda.
> Jika Anda perlu berbagi aliran antara operasi SDK dan kode Anda, bungkus dalam instance `GuzzleHttp\Psr7\Stream` sebelum memasukkannya sebagai parameter perintah. SDK menggunakan aliran, sehingga kode Anda perlu memperhitungkan pergerakan kursor internal aliran. Aliran Guzzle memanggil `fclose` sumber daya aliran yang mendasarinya ketika dihancurkan oleh pengumpul sampah PHP, jadi Anda tidak perlu menutup sendiri aliran tersebut.

Dekorator aliran

Guzzle menyediakan beberapa dekorator aliran yang dapat Anda gunakan untuk mengontrol bagaimana SDK dan Guzzle berinteraksi dengan sumber daya streaming yang disediakan sebagai parameter input ke perintah. Dekorator ini dapat memodifikasi bagaimana penangan dapat membaca dan mencari pada aliran tertentu. Berikut ini adalah daftar sebagian; lebih banyak dapat ditemukan di repositori GuzzleHttpPsr 7.

AppendStream

GuzzleHttp\ Psr7\ AppendStream

Membaca dari beberapa aliran, satu demi satu.

```
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

Digunakan untuk memungkinkan pencarian lebih dari byte yang dibaca sebelumnya pada aliran yang tidak dapat dicari. Ini dapat berguna ketika mentransfer badan entitas yang tidak dapat dicari gagal karena perlu memundurkan aliran (misalnya, yang dihasilkan dari pengalihan). Data yang dibaca dari aliran jarak jauh di-buffer dalam aliran temp PHP sehingga byte yang dibaca sebelumnya di-cache terlebih dahulu di memori, kemudian pada disk.

```
use GuzzleHttp\Psr7;

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

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

```
// 1024

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

InflateStream

GuzzleHttp\ Psr7\ InflateStream

Menggunakan filter zlib.inflate PHP untuk mengembang atau mengempiskan konten gzip.

Dekorator aliran ini melewatkan 10 byte pertama dari aliran yang diberikan untuk menghapus header gzip, mengonversi aliran yang disediakan menjadi sumber daya aliran PHP, dan kemudian menambahkan filter zlib.inflate. Aliran kemudian dikonversi kembali ke sumber daya aliran Guzzle untuk digunakan sebagai aliran Guzzle.

LazyOpenStream

GuzzleHttp\ Psr7\ LazyOpenStream

Dengan malas membaca atau menulis ke file yang dibuka hanya setelah operasi I/O berlangsung di aliran.

```
use GuzzleHttp\Psr7;

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

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

LimitStream

GuzzleHttp\ Psr7\ LimitStream

Digunakan untuk membaca subset atau sepotong objek aliran yang ada. Ini dapat berguna untuk memecah file besar menjadi potongan-potongan kecil untuk dikirim dalam potongan (misalnya, Amazon S3 Multipart Upload API).

```
use GuzzleHttp\Psr7;
```

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

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

NoSeekStream

GuzzleHttp\ Psr7\ NoSeekStream

Membungkus aliran dan tidak memungkinkan pencarian.

```
use GuzzleHttp\Psr7;

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

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

PumpStream

GuzzleHttp\ Psr7\ PumpStream

Menyediakan aliran hanya-baca yang memompa data dari PHP callable.

Saat memanggil callable yang disediakan, PumpStream meneruskan jumlah data yang diminta untuk dibaca ke callable. Callable dapat memilih untuk mengabaikan nilai ini dan mengembalikan lebih sedikit atau lebih byte dari yang diminta. Setiap data tambahan yang dikembalikan oleh callable yang disediakan di-buffer secara internal hingga terkuras menggunakan fungsi read () dari. PumpStream Callable yang disediakan HARUS mengembalikan false ketika tidak ada lagi data untuk dibaca.

Menerapkan dekorator aliran

Membuat dekorator aliran sangat mudah berkat GuzzleHttp\ Psr7\. StreamDecoratorTrait Sifat ini menyediakan metode yang `Psr\Http\Message\StreamInterface` diimplementasikan dengan proxy ke aliran yang mendasarinya. Hanya `StreamDecoratorTrait` dan `use` terapkan metode kustom Anda.

Sebagai contoh, katakanlah kita ingin memanggil fungsi tertentu setiap kali byte terakhir dibaca dari aliran. Ini dapat diimplementasikan dengan mengesampingkan `read()` metode.

```
use Psr\Http\Message\StreamInterface;
use GuzzleHttp\Psr7\StreamDecoratorTrait;

class EofCallbackStream implements StreamInterface
{
    use StreamDecoratorTrait;

    private $callback;

    public function __construct(StreamInterface $stream, callable $cb)
    {
        $this->stream = $stream;
        $this->callback = $cb;
    }

    public function read($length)
    {
        $result = $this->stream->read($length);

        // Invoke the callback when EOF is hit
        if ($this->eof()) {
            call_user_func($this->callback);
        }

        return $result;
    }
}
```

Dekorator ini dapat ditambahkan ke aliran yang ada dan digunakan seperti ini.

```
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!"
```

Menggunakan hasil paginasi di AWS SDK for PHP Versi 3

Beberapa operasi AWS layanan diberi paginasi dan merespons dengan hasil terpotong. Misalnya, `ListObjects` operasi Amazon S3 hanya mengembalikan hingga 1.000 objek sekaligus. Operasi seperti ini (biasanya diawali dengan “list” atau “describe”) memerlukan membuat permintaan berikutnya dengan parameter token (atau marker) untuk mengambil seluruh rangkaian hasil.

Paginator adalah fitur AWS SDK for PHP yang bertindak sebagai abstraksi atas proses ini untuk memudahkan pengembang menggunakan paginated. APIs Sebuah paginator pada dasarnya adalah iterator hasil. Mereka dibuat melalui `getPaginator()` metode klien. Ketika Anda memanggil`getPaginator()`, Anda harus memberikan nama operasi dan argumen operasi (dengan cara yang sama Anda lakukan ketika Anda menjalankan operasi). Anda dapat mengulangi objek paginator menggunakan `foreach` untuk mendapatkan objek individu. `Aws\Result`

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

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

Objek paginator

Objek yang dikembalikan oleh `getPaginator()` metode adalah contoh dari `Aws\ResultPaginator` kelas. Kelas ini mengimplementasikan `iterator` antarmuka asli PHP, itulah sebabnya ia bekerja dengan`foreach`. Hal ini juga dapat digunakan dengan fungsi iterator, seperti`iterator_to_array`, dan terintegrasi dengan baik dengan iterator SPL seperti objek. `LimitIterator`

Objek paginator hanya menyimpan satu “halaman” hasil pada satu waktu dan dieksekusi dengan malas. Ini berarti bahwa mereka hanya membuat permintaan sebanyak yang mereka butuhkan untuk menghasilkan halaman hasil saat ini. Misalnya, `ListObjects` operasi Amazon S3 hanya mengembalikan hingga 1.000 objek sekaligus, jadi jika bucket Anda memiliki ~10.000 objek, paginator perlu melakukan total 10 permintaan. Ketika Anda mengulangi hasil, permintaan pertama dijalankan ketika Anda mulai iterasi, yang kedua dalam iterasi kedua dari loop, dan seterusnya.

Menghitung data dari hasil

Objek Paginator memiliki metode bernama`search()`, yang memungkinkan Anda untuk membuat iterator untuk data dalam satu set hasil. Saat Anda memanggil`search()`, berikan JMESPath ekspresi untuk menentukan data apa yang akan diekstrak. Memanggil `search()` mengembalikan iterator yang menghasilkan hasil ekspresi pada setiap halaman hasil. Ini dievaluasi dengan malas, saat Anda mengulangi iterator yang dikembalikan.

Contoh berikut ini setara dengan contoh kode sebelumnya, tetapi menggunakan `ResultPaginator::search()` metode ini agar lebih ringkas.

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

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

JMESPath ekspresi memungkinkan Anda untuk melakukan hal-hal yang cukup kompleks. Misalnya, jika Anda ingin mencetak semua kunci objek dan awalan umum (yaitu, melakukan ember), Anda dapat melakukan hal berikut. `ls`

```
// List all prefixes ("directories") and objects ("files") in the bucket
```

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

$expression = '[CommonPrefixes[].Prefix, Contents[].Key][]';
foreach ($results->search($expression) as $item) {
    echo $item . "\n";
}
```

pagination asinkron

Anda dapat mengulangi hasil paginator secara asinkron dengan menyediakan callback untuk metode. `each()` `Aws\ResultPaginator` Callback dipanggil untuk setiap nilai yang dihasilkan oleh paginator.

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

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

> **Note**
>
> Menggunakan `each()` metode ini memungkinkan Anda untuk melakukan paginasi pada hasil operasi API sambil secara bersamaan mengirim permintaan lain secara asinkron.

Nilai pengembalian non-null dari callback akan dihasilkan oleh janji berbasis coroutine yang mendasarinya. Ini berarti Anda dapat mengembalikan janji dari panggilan balik yang harus diselesaikan sebelum melanjutkan iterasi atas item yang tersisa, pada dasarnya menggabungkan janji lain ke iterasi. Nilai non-null terakhir yang dikembalikan oleh callback adalah hasil yang memenuhi janji untuk setiap janji hilir. Jika nilai pengembalian terakhir adalah janji, resolusi janji itu adalah hasil yang memenuhi atau menolak janji hilir.

```
// Delete all keys that end with "Foo"
$promise = $results->each(function ($result) use ($s3Client) {
    if (substr($result['Key'], -3) === 'Foo') {
```

```
        // Merge this promise into the iterator
        return $s3Client->deleteAsync([
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'Foo'
        ]);
    }
});

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

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

Pelayan di Versi 3 AWS SDK for PHP

Pelayan membantu membuatnya lebih mudah untuk bekerja dengan sistem yang pada akhirnya konsisten dengan menyediakan cara abstrak untuk menunggu sampai sumber daya masuk ke keadaan tertentu dengan melakukan polling sumber daya. Anda dapat menemukan daftar pelayan yang didukung oleh klien dengan melihat dokumentasi API untuk satu versi klien layanan. Untuk menavigasi ke sana, buka halaman klien di dokumentasi API dan arahkan ke nomor versi tertentu (diwakili oleh tanggal) dan gulir ke bawah ke bagian 'Pelayan'. Tautan ini akan membawa Anda ke bagian pelayan S3.

Dalam contoh berikut, klien Amazon S3 digunakan untuk membuat bucket. Kemudian pelayan digunakan untuk menunggu sampai ember ada.

```
// Create a bucket
$s3Client->createBucket(['Bucket' => 'amzn-s3-demo-bucket']);

// Wait until the created bucket is available
$s3Client->waitUntil('BucketExists', ['Bucket' => 'amzn-s3-demo-bucket']);
```

Jika pelayan harus melakukan polling terlalu sering, itu akan memberikan `\RuntimeException` pengecualian.

Konfigurasi pelayan

Pelayan didorong oleh serangkaian opsi konfigurasi asosiatif. Semua opsi yang digunakan oleh pelayan tertentu memiliki nilai default, tetapi mereka dapat diganti untuk mendukung strategi menunggu yang berbeda.

Anda dapat memodifikasi opsi konfigurasi pelayan dengan meneruskan array `@waiter` opsi asosiatif ke `$args` argumen klien `waitUntil()` dan `getWaiter()` metode.

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

keterlambatan (int)

Jumlah detik untuk menunda antara upaya pemungutan suara. Setiap pelayan memiliki nilai `delay` konfigurasi default, tetapi Anda mungkin perlu mengubah pengaturan ini untuk kasus penggunaan tertentu.

maxAttempt (int)

Jumlah maksimum upaya pemungutan suara untuk diterbitkan sebelum gagal pelayan. Opsi ini memastikan bahwa Anda tidak menunggu sumber daya tanpa batas waktu. Setiap pelayan memiliki nilai `maxAttempts` konfigurasi default, tetapi Anda mungkin perlu mengubah pengaturan ini untuk kasus penggunaan tertentu.

initDelay (int)

Jumlah waktu dalam hitungan detik untuk menunggu sebelum upaya pemungutan suara pertama. Ini mungkin berguna ketika menunggu sumber daya yang Anda tahu akan memakan waktu beberapa saat untuk masuk ke keadaan yang diinginkan.

sebelum (dapat dipanggil)

Fungsi callable PHP yang dipanggil sebelum setiap upaya. Callable dipanggil dengan `Aws\CommandInterface` perintah yang akan dieksekusi dan jumlah upaya yang telah dieksekusi

sejauh ini. Penggunaan `before` callable mungkin untuk memodifikasi perintah sebelum dijalankan atau memberikan informasi kemajuan.

```
use Aws\CommandInterface;

$s3Client->waitUntil('BucketExists', [
    'Bucket'  => 'amzn-s3-demo-bucket',
    '@waiter' => [
        'before' => function (CommandInterface $command, $attempts) {
            printf(
                "About to send %s. Attempt %d\n",
                $command->getName(),
                $attempts
            );
        }
    ]
]);
```

Menunggu secara asinkron

Selain menunggu secara serempak, Anda dapat meminta pelayan untuk menunggu secara asinkron saat mengirim permintaan lain atau menunggu beberapa sumber daya sekaligus.

Anda dapat mengakses janji pelayan dengan mengambil pelayan dari klien menggunakan metode klien. `getWaiter($name, array $args = [])` Gunakan `promise()` metode pelayan untuk memulai pelayan. Janji pelayan dipenuhi dengan `Aws\CommandInterface` yang terakhir yang dieksekusi di pelayan, dan ditolak dengan kesalahan`RuntimeException`.

```
use Aws\CommandInterface;

$waiterName = 'BucketExists';
$waiterOptions = ['Bucket' => 'amzn-s3-demo-bucket'];

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

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

// Call methods when the promise is resolved.
$promise
```

```
    ->then(function () {
        echo "Waiter completed\n";
    })
    ->otherwise(function (\Exception $e) {
        echo "Waiter failed: " . $e . "\n";
    });

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

Mengekspos API pelayan berbasis janji memungkinkan beberapa kasus penggunaan overhead yang kuat dan relatif rendah. Misalnya, bagaimana jika Anda ingin menunggu banyak sumber daya, dan melakukan sesuatu dengan pelayan pertama yang berhasil diselesaikan?

```
use Aws\CommandInterface;

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

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

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

JMESPath ekspresi dalam AWS SDK for PHP Versi 3

JMESPathmemungkinkan Anda untuk secara deklaratif menentukan cara mengekstrak elemen dari dokumen JSON. Ini AWS SDK for PHP memiliki ketergantungan pada jmespath.php untuk memberi daya pada beberapa abstraksi tingkat tinggi seperti Paginator di AWS SDK for PHP Versi 3 dan

Pelayan di AWS SDK for PHP Versi 3, tetapi juga mengekspos pencarian di dan. JMESPath `Aws\ResultInterface` `Aws\ResultPaginator`

Anda dapat bermain-main JMESPath di browser Anda dengan mencoba JMESPath contoh online. Anda dapat mempelajari lebih lanjut tentang bahasa, termasuk ekspresi dan fungsi yang tersedia, dalam JMESPath spesifikasi.

AWS CLIDukungan JMESPath. Ekspresi yang Anda tulis untuk output CLI 100 persen kompatibel dengan ekspresi yang ditulis untuk. AWS SDK for PHP

Mengekstrak data dari hasil

`Aws\ResultInterface`Antarmuka memiliki `search($expression)` metode yang mengekstrak data dari model hasil berdasarkan JMESPath ekspresi. Menggunakan JMESPath ekspresi untuk menanyakan data dari objek hasil dapat membantu menghapus kode bersyarat boilerplate, dan lebih ringkas mengekspresikan data yang sedang diekstraksi.

Untuk mendemonstrasikan cara kerjanya, kita akan mulai dengan output JSON default di bawah ini, yang menjelaskan dua volume Amazon Elastic Block Store (Amazon EBS) Elastic Block Store (Amazon EBS) yang dilampirkan ke instance Amazon yang terpisah. EC2

```
$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"
        }
    }
}
```

Pertama, kita hanya dapat mengambil volume pertama dari daftar Volume dengan perintah berikut.

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

Sekarang, kita menggunakan `wildcard-index` ekspresi `[*]` untuk mengulangi seluruh daftar dan juga mengekstrak dan mengganti nama tiga elemen: `VolumeId` diganti namanya menjadi, diganti

namanya menjadiID, AvailabilityZone dan tetap. AZ Size Size Kita dapat mengekstrak dan mengganti nama elemen-elemen ini menggunakan multi-hash ekspresi yang ditempatkan setelah wildcard-index ekspresi.

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

Ini memberi kita array data PHP seperti berikut:

```
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)
  }
}
```

Dalam multi-hash notasi, Anda juga dapat menggunakan kunci berantai seperti key1.key2[0].key3 untuk mengekstrak elemen yang sangat bersarang di dalam struktur. Contoh berikut menunjukkan ini dengan Attachments[0].InstanceId kunci, alias sederhana. InstanceId (Dalam kebanyakan kasus, JMESPath ekspresi akan mengabaikan spasi.)

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

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

Ekspresi sebelumnya akan menampilkan data berikut:

```
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)
  }
}
```

Anda juga dapat memfilter beberapa elemen dengan `multi-list` ekspresi:`[key1, key2]`. Ini memformat semua atribut yang difilter ke dalam satu daftar berurutan per objek, apa pun jenisnya.

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

Menjalankan pencarian sebelumnya menghasilkan data berikut:

```
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)
  }
}
```

Gunakan `filter` ekspresi untuk memfilter hasil berdasarkan nilai bidang tertentu. Contoh query berikut hanya menghasilkan volume di `us-west-2a` Availability Zone.

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

JMESPath juga mendukung ekspresi fungsi. Katakanlah Anda ingin menjalankan kueri yang sama seperti di atas, tetapi mengambil semua volume di mana volume berada di AWS Wilayah yang dimulai dengan “us-”. Ekspresi berikut menggunakan `starts_with` fungsi, meneruskan string literal dari`us-`. Hasil fungsi ini kemudian dibandingkan dengan nilai literal JSON`true`, hanya meneruskan hasil predikat filter yang dikembalikan `true` melalui proyeksi filter.

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

Mengekstrak data dari paginator

Seperti yang Anda ketahui dari Paginator dalam panduan AWS SDK for PHP Versi 3, `Aws\ResultPaginator` objek digunakan untuk menghasilkan hasil dari operasi API pageable. AWS SDK for PHP Ini memungkinkan Anda untuk mengekstrak dan mengulangi data yang difilter dari `Aws\ResultPaginator` objek, pada dasarnya menerapkan peta datar di atas iterator di mana hasil JMESPath ekspresi adalah fungsi peta.

Katakanlah Anda ingin membuat sebuah `iterator` yang hanya menghasilkan objek dari ember yang lebih besar dari 1 MB. Ini dapat dicapai dengan terlebih dahulu membuat `ListObjects` paginator dan kemudian menerapkan `search()` fungsi ke paginator, membuat iterator yang dipetakan datar di atas data paginasi.

```
$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);
}
```

Menelepon Layanan AWS dari AWS SDK for PHP Versi 3

Bagian berikut berisi contoh, tutorial, tugas, dan panduan yang menunjukkan cara menggunakan layanan AWS SDK for PHP untuk bekerja dengan AWS layanan.

Topik

- Gunakan fitur dan opsi AWS SDK for PHP Versi 3
- Contoh kode dengan panduan untuk AWS SDK for PHP

Gunakan fitur dan opsi AWS SDK for PHP Versi 3

AWS SDK for PHP Versi 3 menyediakan dukungan untuk fitur tambahan dan opsi untuk bekerja dengan Layanan AWS APIs. Bagian dalam topik ini mencakup opsi ini berdasarkan layanan.

Topik

- Menggunakan penangan sesi DynamoDB dengan Versi 3 AWS SDK for PHP
- Fitur dan opsi Amazon S3 di Versi 3 AWS SDK for PHP

Menggunakan penangan sesi DynamoDB dengan Versi 3 AWS SDK for PHP

DynamoDB Session Handler adalah penangan sesi kustom untuk PHP yang memungkinkan pengembang menggunakan Amazon DynamoDB sebagai toko sesi. Menggunakan DynamoDB untuk penyimpanan sesi mengurangi masalah yang terjadi dengan penanganan sesi dalam aplikasi web terdistribusi dengan memindahkan sesi dari sistem file lokal dan ke lokasi bersama. DynamoDB cepat, terukur, mudah diatur, dan menangani replikasi data Anda secara otomatis.

DynamoDB Session Handler menggunakan `session_set_save_handler()` fungsi untuk menghubungkan operasi DynamoDB ke fungsi sesi asli PHP untuk memungkinkan penurunan penggantian yang sebenarnya. Ini termasuk dukungan untuk fitur seperti penguncian sesi dan pengumpulan sampah, yang merupakan bagian dari penangan sesi default PHP.

Untuk informasi selengkapnya tentang layanan DynamoDB, lihat beranda Amazon DynamoDB.

Penggunaan dasar

Langkah 1: Daftarkan handler

Pertama, buat instance dan daftarkan session handler.

```
use Aws\DynamoDb\SessionHandler;

$dynamoDb = new Aws\DynamoDb\DynamoDbClient([
    'region'=>'us-east-1'  // Since version 3.277.10 of the SDK,
]);                        // the 'version' parameter defaults to 'latest'.

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

$sessionHandler->register();
```

Langkah 2. Buat tabel untuk menyimpan sesi Anda

Sebelum Anda benar-benar dapat menggunakan pengendali sesi, Anda perlu membuat tabel untuk menyimpan sesi. Anda dapat melakukan ini sebelumnya dengan menggunakan AWS Konsol untuk Amazon DynamoDB, atau dengan menggunakan. AWS SDK for PHP

Saat membuat tabel ini gunakan 'id' sebagai nama kunci utama. Juga disarankan untuk menyiapkan atribut Time To Live menggunakan atribut 'expires' untuk mendapatkan keuntungan dari pengumpulan sesi sampah otomatis.

Langkah 3. Gunakan sesi PHP seperti biasanya

Setelah penangan sesi terdaftar dan tabel ada, Anda dapat menulis dan membaca dari sesi menggunakan `$_SESSION` superglobal, seperti yang biasa Anda lakukan dengan penangan sesi default PHP. DynamoDB Session Handler merangkum dan mengabstraksi interaksi dengan DynamoDB dan memungkinkan Anda untuk hanya menggunakan fungsi sesi asli PHP dan antarmuka.

```
// 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();
```

Konfigurasi

Anda dapat mengonfigurasi perilaku pengendali sesi menggunakan opsi berikut. Semua opsi bersifat opsional, tetapi pastikan untuk memahami apa defaultnya.

`table_name`

Nama tabel DynamoDB untuk menyimpan sesi. Ini default ke. `'sessions'`

`hash_key`

Nama kunci hash dalam tabel sesi DynamoDB. Ini default ke. `'id'`

`data_attribute`

Nama atribut dalam tabel sesi DynamoDB tempat data sesi disimpan. Ini default ke. `'data'`

`data_attribute_type`

Jenis atribut dalam tabel sesi DynamoDB di mana data sesi disimpan. Ini default ke`'string'`, tetapi secara opsional dapat diatur ke. `'binary'`

`session_lifetime`

Masa pakai sesi yang tidak aktif sebelum itu harus menjadi sampah yang dikumpulkan. Jika tidak disediakan, nilai seumur hidup aktual yang akan digunakan adalah`ini_get('session.gc_maxlifetime')`.

`session_lifetime_attribute`

Nama atribut dalam tabel sesi DynamoDB di mana waktu kedaluwarsa sesi disimpan. Ini default ke. `'expires'`

`consistent_read`

Apakah pengendali sesi harus menggunakan pembacaan yang konsisten untuk `GetItem` operasi. Nilai default-nya `true`.

`locking`

Apakah akan menggunakan penguncian sesi. Nilai default-nya `false`.

`batch_config`

Konfigurasi yang digunakan untuk menghapus batch selama pengumpulan sampah. Opsi ini diteruskan langsung ke objek DynamoDB WriteRequestBatch. Memicu pengumpulan sampah secara manual melalui`SessionHandler::garbageCollect()`.

`max_lock_wait_time`

Waktu maksimum (dalam detik) bahwa pengendali sesi harus menunggu untuk mendapatkan kunci sebelum menyerah. Default untuk adalah `10` dan hanya digunakan dengan penguncian sesi.

`min_lock_retry_microtime`

Waktu minimum (dalam mikrodetik) bahwa pengendali sesi harus menunggu di antara upaya untuk mendapatkan kunci. Defaultnya adalah `10000` dan hanya digunakan dengan penguncian sesi.

`max_lock_retry_microtime`

Waktu maksimum (dalam mikrodetik) bahwa pengendali sesi harus menunggu di antara upaya untuk memperoleh kunci. Defaultnya adalah `50000` dan hanya digunakan dengan penguncian sesi.

Untuk mengonfigurasi Session Handler, tentukan opsi konfigurasi saat Anda membuat instance handler. Kode berikut adalah contoh dengan semua opsi konfigurasi yang ditentukan.

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

Penetapan harga

Selain biaya penyimpanan data dan transfer data, biaya yang terkait dengan penggunaan DynamoDB dihitung berdasarkan kapasitas throughput yang disediakan pada tabel Anda (lihat detail harga Amazon DynamoDB). Throughput diukur dalam satuan kapasitas tulis dan kapasitas baca. Beranda Amazon DynamoDB mengatakan:

Satuan kapasitas baca mewakili satu pembacaan yang sangat konsisten per detik (atau dua pembacaan yang akhirnya konsisten per detik) untuk item sebesar 4 KB. Satuan kapasitas tulis mewakili satu tulis per detik untuk item sebesar 1 KB.

Pada akhirnya, throughput dan biaya yang diperlukan untuk tabel sesi Anda akan berkorelasi dengan lalu lintas dan ukuran sesi yang Anda harapkan. Tabel berikut menjelaskan jumlah operasi baca dan tulis yang dilakukan pada tabel DynamoDB Anda untuk setiap fungsi sesi.

Baca melalui `session_start()`	• 1 operasi baca (hanya 0,5 jika `consistent_read` ada `false`). • (Bersyarat) 1 operasi tulis untuk menghapus sesi jika sudah kedaluwarsa.
Baca melalui `session_start()` (Menggunakan penguncian sesi)	• Minimal 1 operasi tulis. • (Bersyarat) Operasi penulisan tambahan untuk setiap upaya memperoleh kunci pada sesi. Berdasarkan waktu tunggu kunci yang dikonfigurasi dan opsi coba lagi. • (Bersyarat) 1 operasi tulis untuk menghapus sesi jika sudah kedaluwarsa.
Menulis melalui `session_write_close()`	• 1 operasi tulis.
Hapus melalui `session_destroy()`	• 1 operasi tulis.
Pengumpulan Sampah	• 0,5 operasi baca per 4 KB data dalam tabel untuk memindai sesi kedaluwarsa. • 1 operasi tulis per item kedaluwarsa untuk menghapusnya.

Penguncian sesi

DynamoDB Session Handler mendukung penguncian sesi pesimis untuk meniru perilaku pengendali sesi default PHP. Secara default, DynamoDB Session Handler memiliki fitur ini dimatikan karena dapat menjadi hambatan kinerja dan menaikkan biaya, terutama ketika aplikasi mengakses sesi saat menggunakan permintaan Ajax atau iframe. Pertimbangkan dengan cermat apakah aplikasi Anda memerlukan penguncian sesi sebelum mengaktifkannya.

Untuk mengaktifkan penguncian sesi, atur `'locking'` opsi ke `true` saat Anda membuat instance. `SessionHandler`

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

Pengumpulan sampah

Siapkan atribut TTL di tabel DynamoDB Anda, menggunakan atribut 'kedaluwarsa'. Ini akan secara otomatis mengumpulkan sampah sesi Anda dan menghindari kebutuhan untuk mengumpulkan sampah sendiri.

Atau, DynamoDB Session Handler mendukung pengumpulan sampah sesi dengan menggunakan serangkaian dan operasi. `Scan` `BatchWriteItem` Karena sifat dari bagaimana `Scan` operasi bekerja, dan untuk menemukan semua sesi kedaluwarsa dan menghapusnya, proses pengumpulan sampah dapat memerlukan banyak throughput yang disediakan.

Untuk alasan ini, kami tidak mendukung pengumpulan sampah otomatis. Praktik yang lebih baik adalah menjadwalkan pengumpulan sampah terjadi selama waktu off-peak ketika ledakan throughput yang dikonsumsi tidak akan mengganggu sisa aplikasi. Misalnya, Anda dapat memiliki pekerjaan cron malam memicu skrip untuk menjalankan pengumpulan sampah. Skrip ini perlu melakukan sesuatu seperti berikut ini.

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

```
        }
    ]
]);

$sessionHandler->garbageCollect();
```

Anda juga dapat menggunakan `'before'` opsi di dalam `'batch_config'` untuk memperkenalkan penundaan pada `BatchWriteItem` operasi yang dilakukan oleh proses pengumpulan sampah. Ini akan meningkatkan jumlah waktu yang dibutuhkan pengumpulan sampah untuk menyelesaikan, tetapi ini dapat membantu Anda menyebarkan permintaan yang dibuat oleh DynamoDB Session Handler untuk membantu Anda tetap dekat dengan atau dalam kapasitas throughput yang disediakan selama pengumpulan sampah.

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

$sessionHandler->garbageCollect();
```

Praktik terbaik

1. Buat tabel sesi Anda di AWS Wilayah yang secara geografis paling dekat dengan atau di Wilayah yang sama dengan server aplikasi Anda. Ini memastikan latensi terendah antara aplikasi Anda dan database DynamoDB.
2. Pilih kapasitas throughput yang disediakan dari tabel sesi Anda dengan hati-hati. Pertimbangkan lalu lintas yang diharapkan ke aplikasi Anda dan ukuran sesi yang diharapkan. Atau gunakan mode Read/Write kapasitas 'On Demand' untuk meja Anda.
3. Pantau throughput yang Anda konsumsi melalui AWS Management Console atau dengan Amazon CloudWatch, dan sesuaikan pengaturan throughput Anda sesuai kebutuhan untuk memenuhi permintaan aplikasi Anda.
4. Jaga ukuran sesi Anda kecil (idealnya kurang dari 1 KB). Sesi kecil berkinerja lebih baik dan membutuhkan kapasitas throughput yang lebih sedikit.
5. Jangan gunakan penguncian sesi kecuali aplikasi Anda membutuhkannya.

6. Alih-alih menggunakan pemicu pengumpulan sampah sesi bawaan PHP, jadwalkan pengumpulan sampah Anda melalui pekerjaan cron, atau mekanisme penjadwalan lainnya, untuk dijalankan selama jam-jam sibuk. Gunakan `'batch_config'` opsi untuk keuntungan Anda.

Izin IAM yang diperlukan

Untuk menggunakan SessionHhandler DynamoDB, kredenal yang dikonfigurasi harus memiliki izin untuk menggunakan tabel DynamoDB yang Anda buat pada langkah sebelumnya. Kebijakan IAM berikut berisi izin minimum yang Anda butuhkan. Untuk menggunakan kebijakan ini, ganti nilai Resource dengan Amazon Resource Name (ARN) dari tabel yang Anda buat sebelumnya. Untuk informasi selengkapnya tentang membuat dan melampirkan kebijakan IAM, lihat Mengelola Kebijakan IAM di Panduan Pengguna IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SessionHandler",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:Scan",
                "dynamodb:BatchWriteItem"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-
name"
        }
    ]
}
```

Fitur dan opsi Amazon S3 di Versi 3 AWS SDK for PHP

Topik ini membahas fitur dan opsi tambahan yang disediakan oleh AWS SDK for PHP Versi 3 untuk bekerja dengan Amazon S3.

Topik

- Klien Amazon S3 Multi-wilayah dalam Versi 3 AWS SDK for PHP
- Pembungkus aliran Amazon S3 di Versi 3 AWS SDK for PHP
- Transfer file dan direktori
- Enkripsi sisi klien Amazon S3 di Versi 3 AWS SDK for PHP
- Perlindungan integritas data dengan checksum

Klien Amazon S3 Multi-wilayah dalam Versi 3 AWS SDK for PHP

AWS SDK for PHP Versi 3 menyediakan klien multi-wilayah generik yang dapat digunakan dengan layanan apa pun. Ini memungkinkan pengguna untuk menentukan AWS Wilayah mana untuk mengirim perintah dengan memberikan parameter `@region` input ke perintah apa pun. Selain itu, SDK menyediakan klien multi-wilayah untuk Amazon S3 yang merespons secara cerdas kesalahan Amazon S3 tertentu dan mengalihkan perintah yang sesuai. Hal ini memungkinkan pengguna untuk menggunakan klien yang sama untuk berbicara dengan beberapa Wilayah. Ini adalah fitur yang sangat berguna bagi pengguna Amazon S3 Stream Wrapper dengan AWS SDK for PHP Versi 3, yang embernya berada di beberapa Wilayah.

Penggunaan dasar

Pola penggunaan dasar klien Amazon S3 adalah sama apakah menggunakan klien S3 standar atau mitra multi-wilayahnya. Satu-satunya perbedaan penggunaan pada tingkat perintah adalah bahwa AWS Region dapat ditentukan menggunakan parameter `@region` input.

```
// 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**
>
> Saat menggunakan klien Amazon S3 multi-wilayah, Anda tidak akan menemukan pengecualian pengalihan permanen. Klien Amazon S3 standar akan melempar instance `Aws\S3\Exception\PermanentRedirectException` ketika perintah dikirim ke Wilayah yang salah. Klien multi-wilayah akan mengirimkan kembali perintah ke Wilayah yang benar.

Cache Wilayah Bucket

Klien multi-wilayah Amazon S3 memelihara cache internal AWS Wilayah tempat bucket yang diberikan berada. Secara default, setiap klien memiliki cache dalam memori sendiri. Untuk berbagi cache antara klien atau proses, berikan instance `Aws\CacheInterface` sebagai `bucket_region_cache` opsi ke klien multi-wilayah Anda.

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

Pembungkus aliran Amazon S3 di Versi 3 AWS SDK for PHP

Pembungkus aliran Amazon S3 memungkinkan Anda untuk menyimpan dan mengambil data dari Amazon S3 menggunakan fungsi PHP bawaan, seperti,,,,, `file_get_contents` `fopen`, `copy` dan. `rename` `unlink` `mkdir` `rmdir`

Anda perlu mendaftarkan pembungkus aliran Amazon S3 untuk menggunakannya.

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

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

Ini memungkinkan Anda mengakses bucket dan objek yang disimpan di Amazon S3 menggunakan `s3://` protokol. Pembungkus aliran Amazon S3 menerima string yang berisi nama bucket diikuti dengan garis miring ke depan dan kunci objek opsional atau awalan:. `s3://<bucket>[/<key-or-prefix>]`

> **Note**
>
> Pembungkus aliran dirancang untuk bekerja dengan objek dan ember di mana Anda setidaknya memiliki izin baca. Ini berarti bahwa pengguna Anda harus memiliki izin untuk mengeksekusi `ListBucket` pada setiap bucket dan `GetObject` pada objek apa pun yang pengguna perlu berinteraksi. Untuk kasus penggunaan di mana Anda tidak memiliki tingkat izin ini, sebaiknya Anda menggunakan operasi klien Amazon S3 secara langsung.

Unduh data

Anda dapat mengambil isi objek dengan menggunakan`file_get_contents`. Namun, berhati-hatilah dengan fungsi ini; itu memuat seluruh isi objek ke dalam memori.

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

Gunakan `fopen()` saat bekerja dengan file yang lebih besar atau jika Anda perlu melakukan streaming data dari 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**
>
> Kesalahan penulisan file hanya dikembalikan ketika panggilan ke `fflush` dilakukan. Kesalahan ini tidak dikembalikan ketika unflushed dipanggil. `fclose` Nilai pengembalian untuk `fclose` adalah `true` jika menutup aliran, terlepas dari kesalahan apa pun dalam menanggapi `fflush` internalnya. Kesalahan ini juga tidak dikembalikan saat memanggil `file_put_contents` karena bagaimana PHP mengimplementasikannya.

Buka aliran yang dapat dicari

Aliran yang dibuka dalam mode “r” hanya memungkinkan data dibaca dari aliran, dan tidak dapat dicari secara default. Ini agar data dapat diunduh dari Amazon S3 dengan cara yang benar-benar streaming, di mana byte yang dibaca sebelumnya tidak perlu disangga ke dalam memori. Jika Anda membutuhkan aliran agar dapat dicari, Anda dapat meneruskan `seekable` ke opsi konteks aliran suatu fungsi.

```
$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);
}
```

Membuka aliran yang dapat dicari memungkinkan Anda mencari byte yang telah dibaca sebelumnya. Anda tidak dapat melewatkan byte yang belum dibaca dari server jarak jauh. Untuk memungkinkan data yang dibaca sebelumnya diingat, data di-buffer dalam aliran temp PHP menggunakan dekorator aliran. Ketika jumlah data yang di-cache melebihi 2 MB, data dalam aliran temp ditransfer dari memori ke disk. Ingatlah hal ini saat mengunduh file besar dari Amazon S3 menggunakan pengaturan konteks `seekable` streaming.

Unggah data

Anda dapat mengunggah data ke Amazon S3 menggunakan. `file_put_contents()`

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

Anda dapat mengunggah file yang lebih besar dengan streaming data menggunakan `fopen()` dan mode akses aliran “w”, “x”, atau “a”. Pembungkus aliran Amazon S3 tidak mendukung aliran baca dan tulis simultan (misalnya “r+”, “w +”, dll). Ini karena protokol HTTP tidak memungkinkan membaca dan menulis secara simultan.

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

> **Note**
>
> Amazon S3 memerlukan header Content-Length untuk ditentukan sebelum payload permintaan dikirim. Oleh karena itu, data yang akan diunggah dalam suatu `PutObject` operasi di-buffer secara internal menggunakan aliran temp PHP sampai aliran disiram atau ditutup.

> **Note**
>
> Kesalahan penulisan file dikembalikan hanya ketika panggilan ke `fflush` dilakukan. Kesalahan ini tidak dikembalikan ketika unflushed dipanggil. `fclose` Nilai pengembalian untuk `fclose` adalah `true` jika menutup aliran, terlepas dari kesalahan apa pun dalam menanggapi `fflush` internalnya. Kesalahan ini juga tidak dikembalikan saat memanggil `file_put_contents` karena bagaimana PHP mengimplementasikannya.

mode fopen

Fungsi fopen () PHP mengharuskan Anda menentukan `$mode` opsi. Opsi mode menentukan apakah data dapat dibaca atau ditulis ke aliran, dan apakah file harus ada saat membuka aliran.

Pembungkus aliran Amazon S3 mendukung mode berikut untuk aliran yang menargetkan objek Amazon S3.

r	Aliran read-only di mana objek harus sudah ada.
w	Aliran tulis saja. Jika objek sudah ada, itu ditimpa.
a	Aliran tulis saja. Jika objek sudah ada, itu diunduh ke aliran sementara dan penulisan apa pun ke aliran ditambahkan ke data yang diunggah sebelumnya.
x	Aliran tulis saja. Kesalahan muncul jika objek sudah ada.

Fungsi objek lainnya

Stream wrappers memungkinkan banyak fungsi PHP built-in yang berbeda untuk bekerja dengan sistem kustom seperti Amazon S3. Berikut adalah beberapa fungsi yang memungkinkan pembungkus aliran Amazon S3 Anda lakukan dengan objek yang disimpan di Amazon S3.

memutuskan tautan ()	Hapus objek dari ember.
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`
	Anda dapat meneruskan opsi apa pun yang tersedia untuk `DeleteObject` operasi untuk mengubah cara objek dihapus (misalnya menentukan versi objek tertentu).
	`// Delete a specific version of an object from a bucket`
	`unlink('s3://bucket/key', stream_context_create([`
	`'s3' => ['VersionId' => '123']`
	`]);`

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

adalah_file ()	Memeriksa apakah URL adalah file.
	`if (is_file('s3://bucket/key')) {`
	`    echo 'It is a file!';`
	`}`

file_exists ()	Memeriksa apakah ada objek.
	`if (file_exists('s3://bucket/key'))`
	` {`
	`    echo 'It exists!';`
	`}`

tipe file ()	Memeriksa apakah URL memetakan ke file atau bucket (dir).
berkas ()	Muat isi objek dalam array baris. Anda dapat meneruskan opsi apa pun yang tersedia untuk `GetObject` operasi untuk memodifikasi cara file diunduh.
filemtime ()	Dapatkan tanggal modifikasi terakhir dari suatu objek.
ganti nama ()	Ganti nama objek dengan menyalin objek kemudian menghapus aslinya. Anda dapat meneruskan opsi yang tersedia untuk `CopyObject` dan `DeleteObject` operasi ke parameter konteks aliran untuk mengubah cara objek disalin dan dihapus.

> **Note**
>
> Meskipun `copy` umumnya bekerja dengan pembungkus aliran Amazon S3, beberapa kesalahan mungkin tidak dilaporkan dengan benar karena internal fungsi di PHP. `copy` Kami menyarankan Anda menggunakan instance AWSS3 ObjectCopier sebagai gantinya.

Bekerja dengan ember dan folder

Gunakan **`mkdir()`** untuk bekerja dengan ember

Anda dapat membuat dan menelusuri bucket Amazon S3 mirip dengan bagaimana PHP memungkinkan Anda untuk membuat dan melintasi direktori pada sistem file Anda.

Berikut adalah contoh yang membuat ember.

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

> **Note**
>
> Pada bulan April 2023, Amazon S3 secara otomatis mengaktifkan S3 Block Public Access dan menonaktifkan daftar kontrol akses untuk semua bucket yang baru dibuat. Perubahan ini juga memengaruhi cara kerja `mkdir` fungsi dengan izin dan ACLs. `StreamWrapper` Informasi lebih lanjut tersedia di AWS artikel Apa yang Baru dengan ini.

Anda dapat meneruskan opsi konteks aliran ke `mkdir()` metode untuk mengubah cara bucket dibuat menggunakan parameter yang tersedia untuk CreateBucketoperasi.

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

Anda dapat menghapus ember menggunakan `rmdir()` fungsi.

```
// Delete a bucket
```

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

> **Note**
>
> Ember hanya dapat dihapus jika kosong.

Gunakan **`mkdir()`** untuk bekerja dengan folder

Setelah Anda membuat bucket, Anda dapat menggunakan `mkdir()` untuk membuat objek yang berfungsi sebagai folder seperti dalam sistem file.

Cuplikan kode berikut menambahkan objek folder bernama 'my-folder' ke bucket yang ada bernama 'amzn-s3-demo-bucket'. Gunakan karakter forward slash (/) untuk memisahkan nama objek folder dari nama bucket dan nama folder tambahan apa pun.

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

Catatan sebelumnya tentang perubahan izin setelah April 2023 juga ikut bermain saat Anda membuat objek folder. Posting blog ini memiliki informasi tentang cara menyesuaikan izin jika diperlukan.

Gunakan `rmdir()` fungsi untuk menghapus objek folder kosong seperti yang ditunjukkan pada cuplikan berikut.

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

Buat daftar isi ember

Anda dapat menggunakan fungsi PHP opendir (), readdir (), rewinddir (), dan closedir () dengan pembungkus aliran Amazon S3 untuk melintasi isi bucket. Anda dapat meneruskan parameter yang tersedia untuk ListObjectsoperasi sebagai opsi konteks aliran kustom ke `opendir()` fungsi untuk memodifikasi bagaimana objek terdaftar.

```
$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);
}
```

Anda dapat secara rekursif daftar setiap objek dan awalan dalam ember menggunakan PHP. RecursiveDirectoryIterator

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

Cara lain untuk membuat daftar isi bucket secara rekursif yang menimbulkan lebih sedikit permintaan HTTP adalah dengan menggunakan fungsi tersebut. `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";
}
```

Opsi konteks streaming

Anda dapat menyesuaikan klien yang digunakan oleh pembungkus aliran, atau cache yang digunakan untuk menyimpan informasi yang dimuat sebelumnya tentang bucket dan kunci, dengan meneruskan opsi konteks aliran kustom.

Pembungkus aliran mendukung opsi konteks aliran berikut pada setiap operasi.

`client`

`Aws\AwsClientInterface`Objek yang digunakan untuk menjalankan perintah.

`cache`

Contoh `Aws\CacheInterface` untuk digunakan untuk cache statistik file yang diperoleh sebelumnya. Secara default, pembungkus aliran menggunakan cache LRU dalam memori.

Transfer file dan direktori

AWS SDK for PHP Versi 3 menyediakan dua pendekatan untuk mentransfer file dan direktori ke dan dari Amazon S3. Kedua solusi menangani kompleksitas unggahan dan unduhan multibagian untuk file besar, tetapi keduanya berbeda dalam filosofi desain, kumpulan fitur, dan pola penggunaannya.

Ikhtisar opsi transfer

Pilih pendekatan transfer yang paling sesuai dengan kebutuhan aplikasi Anda:

Manajer Transfer S3 (disarankan)

> Perpustakaan modern tingkat tinggi yang menyediakan solusi komprehensif untuk transfer file. Ini menawarkan opsi konfigurasi yang luas, pelacakan kemajuan bawaan, penangan unduhan khusus, dan penanganan kesalahan yang kuat. S3 Transfer Manager menggunakan API berbasis janji dan mendukung operasi file individual dan transfer direktori dengan kemampuan penyaringan tingkat lanjut.

Transfer

> Implementasi transfer direktori yang berfokus secara khusus pada operasi direktori massal. Ini menyediakan API yang lebih sederhana untuk mengunggah dan mengunduh seluruh direktori dengan opsi konfigurasi dasar. Pendekatan ini memiliki lebih sedikit fitur dibandingkan dengan S3 Transfer Manager.

Perbedaan utama

Tabel berikut menyoroti perbedaan utama antara dua pendekatan transfer:

Fitur	Manajer Transfer S3	Transfer
Operasi file individual	Ya (unggah/unduh file tunggal)	Tidak (hanya operasi direktori)
Operasi direktori	Ya (dengan penyaringan lanjutan)	Ya (transfer direktori dasar)
Pelacakan kemajuan	Built-in dengan pendengar khusus	Terbatas (hanya output debug)
Penangan unduhan khusus	Ya	Tidak

Fitur	Manajer Transfer S3	Transfer
Validasi checksum	Otomatis dengan konfigurasi	Manual (opsi add_content_md5)
Penanganan kesalahan	Komprehensif dengan kebijakan kegagalan	Penanganan berbasis janji dasar
Opsi konfigurasi	Ekstensif (8+ opsi)	Dasar (6 opsi)
Desain API	Objek permintaan/respons	Parameter konstruktor sederhana

Memilih pendekatan yang tepat

Gunakan S3 Transfer Manager saat Anda membutuhkan:

- Operasi unggahan atau pengunduhan file individual
- Pelacakan dan pemantauan kemajuan tingkat lanjut
- Penangan unduhan khusus untuk pemrosesan khusus
- Kebijakan penanganan kesalahan dan coba lagi yang komprehensif
- Kontrol halus atas operasi multibagian
- Operasi direktori dengan logika penyaringan yang kompleks

Gunakan Transfer saat Anda membutuhkan:

- Transfer direktori sederhana-ke/dari-S3
- Konfigurasi dan pengaturan minimal
- Kompatibilitas dengan kode yang ada yang menggunakan Transfer
- Fungsionalitas unggahan multipart dasar

> **Note**
>
> Untuk aplikasi baru, kami sarankan menggunakan S3 Transfer Manager karena memberikan solusi yang lebih komprehensif dan fleksibel untuk transfer file.

Manajer Transfer S3

S3 Transfer Manager menyediakan antarmuka untuk mengunggah dan mengunduh file ke dan dari Amazon S3. Anda dapat menggunakannya untuk operasi file tunggal atau operasi direktori.

Manajer Transfer menangani unggahan dan unduhan multibagian secara otomatis. Ini mengelola file besar dan melacak kemajuan. Pustaka mengimplementasikan praktik terbaik untuk transfer data S3. Anda dapat menggunakannya untuk membangun kemampuan transfer file dalam aplikasi PHP Anda.

Fitur utama

Manajer Transfer S3 menyediakan fitur-fitur utama berikut:

- API sederhana: Unggah dan unduh file dan direktori
- Penangan unduhan khusus: Menerapkan logika unduhan Anda sendiri
- Unggahan dan unduhan multipart otomatis: Menangani file besar secara otomatis
- Pemrosesan bersamaan: Maksimalkan throughput
- Pelacakan kemajuan: Pantau status transfer
- Perilaku yang dapat disesuaikan: Konfigurasikan opsi ekstensif
- Penanganan kesalahan: Konfigurasikan kebijakan coba ulang dan kegagalan
- Operasi direktori: Transfer banyak file dalam batch
- Validasi Checksum: Pastikan integritas data

Penginstalan

Manajer Transfer S3 termasuk dalam AWS SDK for PHP Versi 3. Anda tidak perlu menginstalnya secara terpisah.

Untuk menginstal menggunakan Composer, jalankan perintah berikut:

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

Perintah ini menginstal AWS SDK for PHP Versi 3 lengkap, termasuk S3 Transfer Manager.

Penggunaan dasar

Contoh berikut menunjukkan cara menggunakan S3 Transfer Manager:

```
<?php
```

```
use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

// Create a transfer manager with default configuration.
$transferManager = new S3TransferManager($s3Client);

// Alternative: Create transfer manager with null client. S3 Transfer Manager uses a
 default S3 client.
$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Example: Upload a file.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/local/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'path/to/s3/file.txt',
        ]
    )
);

// Wait for the upload to complete.
$result = $uploadPromise->wait();

echo "Upload complete!\n";
```

Important

Saat menggunakan S3 Transfer Manager untuk membuat klien Amazon S3 default, pelanggan dapat meneruskan Wilayah default untuk klien menggunakan `default_region`

parameter dalam opsi `config` Manajer Transfer, jika tidak, klien Amazon S3 mencoba menyelesaikan Wilayah menggunakan perilaku default untuk menyelesaikan konfigurasi, dan jika Wilayah tidak diselesaikan maka pengecualian akan dilemparkan.

Membuat manajer transfer

Anda dapat membuat manajer transfer dengan dua cara:

Dengan klien S3 yang ada

Lulus `S3Client`instance yang ada ke `S3TransferManager` *<add link>* konstruktor.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\S3TransferManager;

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

// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

$transferManager = new S3TransferManager($s3Client);
```

Dengan pembuatan klien S3 default

Lulus `null` sebagai klien dan tentukan opsi konfigurasi.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);
```

Konfigurasi

S3 Transfer Manager menerima opsi konfigurasi untuk menyesuaikan perilakunya. Berikan opsi ini saat Anda membuat instance manajer transfer. Parameter konfigurasi dapat berupa array atau instance dari `S3TransferManagerConfig`*<add link>*.

Contoh berikut mengkonfigurasi instance S3 Transfer Manager:

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null,
    [
        // 10MB parts for multipart operations.
        'target_part_size_bytes' => 10 * 1024 * 1024,

        // Use multipart upload for files larger than 20MB.
        'multipart_upload_threshold_bytes' => 20 * 1024 * 1024,

        // Enable checksum calculation for data integrity.
        'request_checksum_calculation' => 'when_supported',

        // Enable checksum validation when getting objects.
        'response_checksum_validation' => 'when_supported',

        // Use part-based multipart downloads.
        'multipart_download_type' => 'part',

        // Allow up to 10 concurrent operations.
        'concurrency' => 10,

        // Enable progress tracking.
        'track_progress' => true,

        // Set default region for default S3 client construction.
        'default_region' => 'us-west-2',
    ]
);
```

> **Note**
>
> Saat Anda menyediakan konfigurasi sebagai array ke`S3TransferManager`, SDK secara internal memanggil `S3TransferManagerConfig::fromArray` untuk mengonversinya ke tipe yang tepat.

Opsi konfigurasi

Semua opsi konfigurasi bersifat opsional dan menggunakan nilai default saat tidak ditentukan.

Opsi	Jenis	Default	Deskripsi
`target_part_size_bytes`	int	8MB	Ukuran bagian minimum untuk unggahan/unduhan multipart.
`multipart_upload_threshold_bytes`	int	16MB	Ambang ukuran file untuk menggunakan unggahan multipart.
`request_checksum_calculation`	string	'ketika_didukung'	Mengaktifkan perhitungan checksum. Nilai yang valid adalah 'when_supported', 'when_required'.
`response_checksum_validation`	string	'ketika_didukung'	Mengaktifkan validasi checksum saat mendapatkan objek. Nilai yang valid adalah 'when_supported', 'when_required'.
`multipart_download_type`	string	'bagian'	Unduh strategi untuk file besar. Nilai yang valid adalah 'bagian' (unduhan multibagian), 'berkisar' (permintaan rentang).
`concurrency`	int	5	Jumlah maksimum operasi bersamaan.

Opsi	Jenis	Default	Deskripsi
`track_progress`	bool	SALAH	Apakah akan melacak kemajuan transfer.
`default_region`	string	'kami-timur-1'	Wilayah AWS untuk digunakan jika tidak ada klien S3 yang disediakan.

Operasi file

S3 Transfer Manager menyediakan metode untuk mengunggah dan mengunduh file individual.

Unggah file lokal

Untuk mengunggah file ke Amazon S3, gunakan metode ini `upload`*<add link>*.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Source can be from a local path to a file.
$source = '/path/to/local/file.txt';
// Or the source can be an instance of StreamInterface.
$source = GuzzleHttp\Psr7\Utils::streamFor('Hello World!');

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        $source,
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'path/to/s3/file.txt',
            // Additional `putObject` parameters as needed.
        ],
        [
            // Optional configuration overrides.
```

```
            'multipart_upload_threshold_bytes' => 100 * 1024 * 1024, // 100MB
            'target_part_size_bytes'           => 10 * 1024 * 1024, // 10MB
            'track_progress'                   => true,
            'request_checksum_calculation'     => 'when_required',
        ]
    )
);

// The upload is asynchronous, you can wait for it to complete.
$result = $uploadPromise->wait();

// Or you can use the promise for more complex workflows.
$result = $uploadPromise->then(
    function ($result) {
        echo "Upload succeeded!";
    },
    function ($error) {
        echo "Upload failed: " . $error->getMessage();
    }
)->wait();
```

Parameter metode **`upload`**

`upload`Metode ini menerima instance dari `UploadRequest` *<add link>* sebagai argumen.

Parameter **`UploadRequest`**

Parameter	Jenis	Diperlukan	Deskripsi
`$source`	senar| `StreamInterface`	Ya	Jalur jalur file lokal atau aliran yang berisi data untuk diunggah.
`$uploadRequestArgs`	array	Ya	Unggah argumen permintaan (harus menyertakan Bucket dan Key).
`$config`	array	Tidak	Penggantian konfigurasi khusus untuk unggahan ini. Untuk informasi selengkapnya tentang opsi konfigurasi, lihat bagian berikut.

Parameter	Jenis	Diperlukan	Deskripsi
`$listeners`	array	Tidak	Array `TransferListener` objek untuk memantau unggahan ini.
`$progress Tracker`	TransferListener	Tidak	Pelacak kemajuan untuk unggahan ini.

`$config`pilihan untuk **`UploadRequest`**

SDK menyelesaikan `$config` nilai default dari konfigurasi S3 Transfer Manager.

Opsi	Jenis	Diperlukan	Deskripsi
`multipart _upload_t hreshold_ bytes`	int	Tidak	Mengganti ambang batas default saat unggahan multibagian dipicu.
`target_pa rt_size_b ytes`	int	Tidak	Mengganti ukuran bagian target default dalam byte.
`track_pro gress`	bool	Tidak	Mengganti opsi default untuk mengaktifkan pelacakan kemajuan. Jika opsi ini `true` dan Anda tidak memberikan `progressTracker` parameter, SDK menggunakan implementasi default.
`concurrency`	int	Tidak	Mengganti nilai default untuk konkurensi.
`request_c hecksum_c alculation`	string	Tidak	Mengganti nilai default untuk apakah perhitungan checksum permintaan harus dilakukan.

Ketika `upload` metode berjalan dengan sukses, ia mengembalikan file `UploadResult`*<add link>*.

Unduh objek S3

Untuk mengunduh objek dari Amazon S3, gunakan metode ini `download`*<add link>*.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Download using an S3 URI.
$downloadPromise = $transferManager->download(
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
        [
            // Additional `getObject` parameters as needed.
        ],
        [
            // Optional configuration overrides.
            'response_checksum_validation' => 'when_required',
            'track_progress'               => true,
            'target_part_size_bytes'       => 10 * 1024 * 1024, // 10MB
        ]
    )
);

// Wait for the download to complete.
$result = $downloadPromise->wait();

// The SDK uses a stream-based download handler by default.
$stream = $result->getDownloadDataResult();

// You can either get the content.
$content = $stream->getContents();

// Or write the stream to a file.
```

```
file_put_contents('/path/to/local/file.txt', $stream);

// Don't forget to close the stream.
$stream->close();
```

Parameter metode **`download`**

`download`Metode ini menerima instance dari `DownloadRequest` *<add link>* sebagai argumen.

Parameter DownloadRequest

Parameter	Jenis	Diperlukan	Deskripsi
`$source`	senar | array| `null`	Tidak	Objek untuk men-download dari S3. Anda dapat memberikan parameter ini sebagai string URI S3 (" s3:// amzn-s3-demo-bucket/key "), array dengan kunci Bucket dan Key, atau`null`. Ketika nilai in`null`, berikan nilai Bucket dan Key dalam `$downloadRequestArgs` parameter.
`$download RequestAr gs`	array	Tidak	Argumen permintaan objek unduhan tambahan. Jika `$source` yan`null`, berikan nilai Bucket dan Key di sini.
`$config`	array	Tidak	Penggantian konfigurasi khusus untuk unduhan ini. Untuk informasi selengkapnya tentang opsi konfigura si, lihat bagian berikut.
`$download Handler`	`AbstractD ownloadHa ndler` *<add link>*|`null`	Tidak	Handler yang menerima setiap potongan objek dan mengelola unduhannya. Handler default untuk `download` metode ini menggunakan handler berbasis streaming. Handler ini

Parameter	Jenis	Diperlukan	Deskripsi
			mendorong setiap potongan objek ke dalam aliran. Handler default untuk `downloadFile` metode (lihatthe section called "Unduh objek S3 ke file lokal") menggunakan handler berbasis file. Handler ini menulis setiap potongan ke file. Anda dapat menerapkan sendiri `DownloadHandler` untuk menyesuaikan di mana dan bagaimana SDK menyimpan data yang diunduh. Misalnya, Anda dapat menyimpan data dalam database, penyimpanan cloud, atau pipeline pemrosesan khusus, bukan file atau aliran.
`$progressTracker`	`TransferListener`	Tidak	Pelacak kemajuan untuk unduhan ini.
`$listeners`	`array`	Tidak	Array `AbstractTransferListener` objek untuk memantau unduhan ini.

`$config`pilihan untuk `DownloadRequest`

SDK menyelesaikan `$config` nilai default dari konfigurasi S3 Transfer Manager.

Opsi	Jenis	Diperlukan	Deskripsi
`multipart_download_type`	string	Tidak	Mengganti jenis unduhan multipart default. Nilai yang valid adalah 'bagian', 'berkisar'

Opsi	Jenis	Diperlukan	Deskripsi
`response_checksum_validation`	string	Tidak	Mengganti nilai yang diselesaikan dari konfigurasi manajer transfer untuk mengetahui apakah validasi checksum harus dilakukan. SDK mempertimbangkan opsi ini hanya jika tidak `ChecksumMode` ada di `$getObjectRequestArgs` file. `DownloadRequest`
`track_progress`	bool	Tidak	Mengganti opsi default untuk mengaktifkan pelacakan kemajuan. Jika opsi ini `true` dan Anda tidak memberikan `progressTracker` parameter dengan SDK menggunakan implementasi default. `DownloadRequest`, Gunakan opsi ini untuk mengaktifkan pelacak kemajuan default saat `$progressTracker` parameter nya`null`.
`target_part_size_bytes`	int	Tidak	Ukuran bagian dalam byte untuk digunakan dalam berbagai unduhan multibagian. Jika Anda tidak memberikan parameter ini, unduhan menggunakan `target_part_size_bytes` konfigurasi pada manajer transfer.

Ketika `download` metode berjalan dengan sukses, ia mengembalikan file `DownloadResult`*<add link>*.

Unduh objek S3 ke file lokal

Untuk mengunduh objek S3 langsung ke file lokal, gunakan `downloadFile` metode ini:

```
<?php

use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$downloadFilePromise = $transferManager->downloadFile(
    new DownloadFileRequest(
        '/path/to/local/file.txt',  // Destination file path.
        false,  // Fail when destination exists.
        new DownloadRequest(
            's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
            [],  // `getObject` request args
            [
                'track_progress'         => true,
                'target_part_size_bytes' => 10 * 1024 * 1024, // 10MB parts
            ]
        )
    )
);

// Wait for download to complete.
$result = $downloadFilePromise->wait();

// `getDownloadDataResult()` returns the string for the file path of the downloaded
 content because
//  the default `DownloadHandler` used by `downloadFile()` is a file-based handler.
echo "File downloaded successfully at {$result->getDownloadDataResult()}!\n";
```

Parameter metode **`downloadFile`**

`downloadFile`Metode ini menerima instance dari `DownloadFileRequest` sebagai argumen.

Parameter **`DownloadFileRequest`**

Opsi	Jenis	Diperlukan	Deskripsi
`$destination`	string	Ya	Jalur lokal tempat file disimpan.
`$failsWhenDestinationExists`	bool	Ya	Apakah akan gagal jika file tujuan sudah ada. Jika opsi ini `false` dan file tujuan ada, SDK menghapus atau mengganti file yang ada.
`$downloadRequest`	`DownloadRequest`	Ya	`DownloadRequest` Objek yang dikonfigurasi. Untuk informasi lebih lanjut, lihat `DownloadRequest` objek.

Ketika `downloadFile` metode berjalan dengan sukses, ia mengembalikan a `DownloadResult`*<add link>*.

Operasi direktori

S3 Transfer Manager dapat menangani seluruh transfer direktori.

Unggah direktori

S3 Transfer Manager dapat mengunggah seluruh direktori ke bucket S3. *<Are the key values the filenames when uploaded?>*

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
```

```
]);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/local/directory',
        'amzn-s3-demo-bucket',
        [
            // Additional `putObject` parameters that apply to all files.
            'ACL'          => 'public-read',
            'CacheControl' => 'max-age=3600',
        ],
        [
            // Configuration options.
            'recursive'                     => true,
            'follow_symbolic_links'         => false,
            's3_prefix'                     => 'uploads/2023/',
            's3_delimiter'                  => '/',
            'track_progress'                => true,
            'filter'                        => function ($file) {
                // Upload only .jpg files.
                return pathinfo($file, PATHINFO_EXTENSION) === 'jpg';
            },
            'upload_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ContentType'] = 'image/jpeg';
            },
        ]
    )
);

// Wait for the upload process to complete.
$result = $uploadDirPromise->wait();

$uploaded = $result->getObjectsUploaded();
$failed = $result->getObjectsFailed();
echo "Uploaded {$uploaded} files. Failed: {$failed}\n";
```

Parameter metode **`uploadDirectory`**

`uploadDirectory`Metode ini menerima instance dari `UploadDirectoryRequest` *<add link>* sebagai argumen.

Parameter **`UploadDirectoryRequest`**

Parameter	Jenis	Diperlukan	Deskripsi
`$sourceDirectory`	string	Ya	Jalur direktori lokal untuk diunggah.
`$targetBucket`	string	Ya	Nama bucket S3 tujuan.
`$uploadRequestArgs`	array	Tidak	Parameter permintaan objek unggahan tambahan untuk semua file.
`$config`	array	Tidak	Opsi konfigurasi untuk upload direktori. Untuk informasi selengkapnya tentang opsi konfigurasi, lihat bagian berikut.
`$listeners`	array	Tidak	Array `TransferListener` objek. SDK mengkloning masing-masing `TransferListener` per file.
`$max_depth`	bool	-1 “Runtime default”	Untuk menunjukkan kedalaman maksimum file rekursif tiga jalan.
`$progressTracker`	`TransferListener`	Tidak	Pelacak kemajuan untuk semua unggahan.

`$config`pilihan untuk **`UploadDirectoryRequest`**

Opsi	Jenis	Default	Deskripsi
`recursive`	bool	`false`	Apakah akan mengunggah subdirektori secara rekursif.

Opsi	Jenis	Default	Deskripsi
`follow_symbolic_links`	bool	`false`	Apakah akan mengikuti tautan simbolis.
`s3_prefix`	string	''	Awalan untuk menambahkan ke semua tombol objek.
`s3_delimiter`	string	'/'	Pembatas untuk digunakan dalam kunci objek S3.
`track_progress`	bool	`false`	Apakah akan melacak kemajuan.
`max_concurrency`	int	100	Jumlah maksimum unggahan bersamaan.
`filter`	dapat dipanggil	null	Berfungsi untuk memfilter file mana yang akan diunggah.
`upload_object_request_modifier`	dapat dipanggil	null	Fungsi untuk menyesuaikan setiap permintaan unggahan.
`failure_policy`	dapat dipanggil	null	Berfungsi untuk menangani kegagalan upload.

Jenis opsi yang dapat dipanggil **`$config`**

Nama opsi	nama parameter	jenis parameter	info parameter
`filter`	`$file`	`SplFileInfo` |string	Jika dilemparkan ke string, itu adalah jalur file. Jika tidak, itu adalah contoh dari`SplFileInfo` .

Nama opsi	nama parameter	jenis parameter	info parameter
`upload_object_request_modifier`	`$requestArgs`	array	Argumen permintaan untuk setiap permintaan upload individu. Untuk informasi selengkapnya tentang opsi array, lihat bagian sintaks parameter dari metode putObject.
`failure_policy`	`$uploadRequestArgs`	array	Argumen permintaan untuk setiap permintaan upload individu. Untuk informasi selengkapnya tentang opsi array, lihat bagian sintaks parameter dari metode putObject untuk opsi array.
	`$uploadDirectoryArgs`	array	Array dengan direktori sumber dan bucket target untuk operasi direktori upload.
	`$reason`	`Throwable` |string	Pemegang pengecualian yang berisi informasi tentang apa yang menyebabkan kegagalan.

Nama opsi	nama parameter	jenis parameter	info parameter
	`$uploadDirectoryResult`	`UploadDirectoryResult`	Objek yang berisi jumlah objek yang berhasil diunggah dan nomor yang gagal.

Ketika `uploadDirectory` metode berjalan dengan sukses, ia mengembalikan file `UploadDirectoryResult`*<add link>*.

Direktori unduhan

Anda dapat mengunduh direktori dari bucket S3. *<Are the filenames the key values when downloaded? If there is no extension in the key, is there also no extension on the file?>*

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/local/directory',
        [
            // Additional `getObject` parameters that apply to all files.
        ],
        [
            // Configuration options.
            's3_prefix'                         => 'uploads/2023/',
            's3_delimiter'                      => '/',
            'track_progress'                    => true,
            'filter'                            => function ($key) {
```

```
                // Download only .jpg files.
                return pathinfo($key, PATHINFO_EXTENSION) === 'jpg';
            },
            'download_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ResponseContentType'] = 'image/jpeg';
            },
            'list_objects_v2_args'              => [
                'MaxKeys' => 1000,
                'Prefix'  => 'uploads/2023/',
            ],
            'fails_when_destination_exists'    => true,
        ]
    )
);

// Wait for the download process to complete.
$result = $downloadDirPromise->wait();

$downloaded = $result->getObjectsDownloaded();
$failed = $result->getObjectsFailed();
echo "Downloaded {$downloaded} files. Failed: {$failed}\n";
```

Parameter metode **`downloadDirectory`**

`downloadDirectory`Metode ini menerima instance dari `DownloadDirectoryRequest` *<add link>* sebagai argumen.

Parameter **`DownloadDirectoryRequest`**

Parameter	Jenis	Diperlukan	Deskripsi
`$sourceBucket`	string	Ya	Sumber nama bucket S3 dari mana objek diunduh.
`$destinationDirectory`	string	Ya	Direktori lokal untuk mengunduh file ke.

Parameter	Jenis	Diperlukan	Deskripsi
`$downloadRequestArgs`	array	Tidak	Parameter permintaan objek unduhan tambahan untuk semua file.
`$config`	array	Tidak	Opsi konfigurasi untuk unduhan direktori. Untuk informasi selengkapnya tentang opsi konfigurasi, lihat bagian berikut.
`$listeners`	array	Tidak	Array `TransferListener` objek. SDK mengkloning masing-masing `TransferListener` per file.
`$progressTracker`	`TransferListener`	Tidak	Pelacak kemajuan untuk semua unduhan.

`$config`pilihan untuk **`DownloadDirectoryRequest`**

Opsi	Jenis	Default	Deskripsi
`s3_prefix`	string	"	Awalan untuk pemfilteran (jika tidak di`list_objects_v2_args`).
`track_progress`	bool	`false`	Apakah akan melacak kemajuan.
`filter`	dapat dipanggil	`null`	Berfungsi untuk memfilter objek S3 mana yang akan diunduh.
`download_object_request_modifier`	dapat dipanggil	`null`	Fungsi untuk menyesuaikan setiap permintaan unduhan.

Opsi	Jenis	Default	Deskripsi
`list_objects_v2_args`	array	[]	Argumen untuk `ListObjectsV2` operasi. Operasi ini mengambil metadata untuk objek yang akan diunduh.
`fails_when_destination_exists`	bool	false	Apakah akan gagal ketika jalur file tujuan objek sudah ada.
`failure_policy`	dapat dipanggil	null	Berfungsi untuk menangani kegagalan unduhan.
`max_concurrency`	int	100	Jumlah maksimum unduhan bersamaan.
`target_part_size_bytes`	int	bervariasi	Ukuran bagian untuk unduhan multibagian saat jenis unduhan multibagian berkisar.

Jenis opsi yang dapat dipanggil **`$config`**

Nama opsi	nama parameter	jenis parameter	info parameter
`filter`	`$objectKey`	string	Nama kunci objek dari ember.
`download_object_request_modifier`	`$requestArgs`	array	Argumen permintaan untuk setiap permintaan unduhan individu. Untuk informasi selengkapnya tentang opsi array, lihat bagian

Nama opsi	nama parameter	jenis parameter	info parameter
			sintaks parameter dari metode getObject.
`failure_policy`	`$downloadObjectRequestArgs`	array	Argumen permintaan untuk setiap permintaan unduhan individu. Untuk informasi selengkapnya tentang opsi array, lihat bagian sintaks parameter dari metode getObject.
	`$downloadDirectoryRequest`	array	Array dengan bucket sumber dan direktori target untuk operasi direktori unduhan.
	`$reason`	`Throwable`	Pemegang pengecualian yang berisi informasi tentang apa yang menyebabkan kegagalan.
	`$downloadDirectoryResult`	DownloadDirectoryResult *<add link>*	Objek yang berisi jumlah objek yang berhasil diunduh dan nomor yang gagal.

Bagaimana operasi direktori bekerja

Bagian ini menjelaskan bagaimana S3 Transfer Manager menangani jalur file dan kunci objek selama operasi direktori.

Upload penanganan jalur

Saat Anda mengunggah direktori, SDK akan membuat kunci objek S3 dari jalur file:

1. Menghitung jalur setiap file relatif terhadap direktori sumber
2. Menambahkan `s3_prefix` nilai jika ditentukan (secara otomatis menambahkan garis miring)
3. Mengganti pemisah direktori OS dengan `s3_delimiter` (default) `/`

`recursive`Opsi mengontrol apakah subdirektori disertakan. Ketika `false` (default), hanya file tingkat atas yang diunggah. Kapan`true`, semua file di subdirektori diunggah, melestarikan struktur direktori dalam kunci objek.

> **Note**
>
> Kunci di S3 untuk setiap file yang diunggah adalah awalan s3 yang disediakan, secara default adalah null, ditambah jalur relatif file yang dimulai dari direktori sumber yang ditentukan. Juga, kami mengganti pemisah direktori dalam nama kunci yang diselesaikan dengan pembatas s3 yang disediakan, yang secara default akan menjadi /.
> Contoh 1
>
> ```
> $sourceDirectory = "/opt/my-upload-directory";
> $s3Prefix = "important/";
> $s3Delimiter = "/"; // Default value
> ```
>
> Struktur Direktori
>
> ```
> /opt/my-upload-directory/
> -- my-file1.txt
> -- my-file-2.txt
> -- sub-dir/
> ---- my-another-file1.txt
> ---- my-another-file2.txt
> ```
>
> Kunci untuk setiap file adalah:
>
> ```
> $s3Prefix + $fileRelativePath;
> ```
>
> ```
> - important/my-file1.txt
> - important/my-file-2.txt
> - important/sub-dir/my-another-file1.txt
> ```

```
- important/sub-dir/my-another-file2.txt
```

Contoh 2

```
$sourceDirectory = "/opt/my-docs";
$s3Prefix = ''; // No provided and it will defaulted to empty string
$s3Delimiter = "/"; // Default value
```

Struktur Direktori

```
/opt/my-docs/
-- my-file1.txt
-- my-file-2.txt
-- sub-dir/
---- my-another-file1.txt
---- my-another-file2.txt
```

Kunci untuk setiap file adalah:

```
$s3Prefix + $fileRelativePath;
```

```
- my-file1.txt
- my-file-2.txt
- sub-dir/my-another-file1.txt
- sub-dir/my-another-file2.txt
```

Example Upload penanganan jalur direktori

```
<?php
// Source directory: /home/user/photos/
//   fla/beach.jpg
//   fla/sunset.jpg
//   fla/pool/party.jpg
// With s3_prefix: '2023' and recursive: true
// Results:
//   2023/fla/beach.jpg
//   2023/fla/sunset.jpg
```

```
//   2023/fla/pool/party.jpg
```

Unduh penanganan jalur

Saat Anda mengunduh direktori, SDK akan membuat jalur file lokal dari kunci objek S3:

1. Menghapus `s3_prefix` dari kunci objek jika ditentukan
2. Mengganti pembatas S3 (`/`) dengan pemisah direktori OS
3. Menambahkan hasil ke direktori tujuan

Download selalu bersifat rekursif. SDK mengambil semua objek di bawah awalan yang ditentukan dan membuat subdirektori sesuai kebutuhan. Ini memvalidasi bahwa jalur tidak menyelesaikan di luar direktori tujuan untuk keamanan.

Example Unduh penanganan jalur direktori

```
<?php
// S3 objects:
//   2023/fla/beach.jpg
//   2023/fla/sunset.jpg
//   2023/fla/pool/party.jpg
// With s3_prefix: '2023' and destination: /home/user/downloads
// Results:
//   /home/user/downloads/fla/beach.jpg
//   /home/user/downloads/fla/sunset.jpg
//   /home/user/downloads/fla/pool/party.jpg
```

> **Note**
>
> Jalur file untuk objek yang diunduh akan diselesaikan sebagai berikut:
>
> - Awalan s3, jika disediakan, akan dihapus dari kunci objek.
> - Jika S3delimiter berbeda dari pemisah direktori OS maka, pembatas s3 akan diganti dengan pemisah direktori OS.
>
> Contoh 1
>
> ```
> $s3Prefix = "my-prefix";
> $s3Objects = [
> ```

```
    "my-prefix/file-1.txt",
    "my-prefix/file-2.txt",
    "my-prefix/subdir/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Setelah benda-benda tersebut diunduh akan ditempatkan sebagai berikut:

```
/opt/bucket/downloads/
-- file-1.txt
-- file-2.txt
-- subdir/
---- file-1.txt
```

Seperti yang bisa kita lihat, awalan telah dihapus dari jalur file akhir untuk objek.
Contoh 2 - tanpa awalan s3

```
$s3Prefix = ""; // No provided
$s3Objects = [
    "README.md",
    "my-docs/file-1.txt",
    "my-docs/file-2.txt",
    "my-docs/statements/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Setelah benda-benda tersebut diunduh akan ditempatkan sebagai berikut:

```
/opt/bucket/downloads/
-- README.md
-- my-docs/
---- file-1.txt
---- file-2.txt
---- statements/
------ file-1.txt
```

Transfer pendengar

Anda dapat menerapkan pendengar Anda sendiri yang bereaksi terhadap peristiwa transfer yang berbeda. Acara tersebut adalah:

- **`transferInitiated`**: Saat transfer dimulai
- **`bytesTransferred`**: Saat transfer transfer byte dari sumber ke tujuan
- **`transferComplete`**: Ketika transfer selesai tanpa kesalahan
- **`transferFail`**: Ketika transfer gagal sebelum selesai

Setiap `S3TransferManager` operasi (misalnya, `uploadFile` dan`downloadDirectory`) memiliki `listeners` parameter. Parameter ini adalah array yang mengharapkan setiap item menjadi instance dari. `AbstractTransferListener`

`AbstractTransferListener`adalah kelas dasar abstrak yang dapat Anda perluas untuk memantau operasi transfer. Ini menyediakan implementasi metode kosong untuk semua peristiwa transfer. Anda hanya perlu mengganti metode untuk acara yang ingin Anda tangani.

Setiap `AbstractTransferListener` metode menerima `$context` argumen. Argumen ini berisi informasi nilai kunci tentang operasi. Konteksnya meliputi:

- `request_args`(array) - Parameter permintaan
- `progress_snapshot`(instance of `TransferProgressSnapshot`*<add link>*) - Status permintaan saat ini
- `reason`(contoh`Throwable`) - disediakan hanya untuk `transferFail` metode

`progress_snapshot`Properti ini juga membawa `reason` properti untuk peristiwa kegagalan. Anda dapat mengakses pengecualian melalui `$context['reason']` atau melalui`$context['progress_snapshot']#getReason()`.

Kode berikut menunjukkan struktur `AbstractTransferListener` kelas:

```
<?php

namespace Aws\S3\S3Transfer\Progress;

abstract class AbstractTransferListener
{
```

```
    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the request initialization.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferInitiated(array $context): void {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function bytesTransferred(array $context): bool {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferComplete(array $context): void {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *    as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     * - reason: (Throwable) The exception originated by the transfer failure.
     *
     * @return void
     */
    public function transferFail(array $context): void {}
}
```

Contoh berikut menunjukkan `AbstractTransferListener` implementasi yang hanya mengesampingkan `transferInitiated` dan `transferFail` metode:

```
<?php

namespace MyApp\Listeners;

use Aws\S3\S3Transfer\Progress\AbstractTransferListener;

class MyCustomListener extends AbstractTransferListener
{
    public function transferInitiated(array $context): void
    {
        $progressSnapshot = $context['progress_snapshot'];

        echo "Transfer initiated for object with identifier: "
            . $progressSnapshot->getIdentifier();
    }

    public function transferFail(array $context): void
    {
        $progressSnapshot = $context['progress_snapshot'];
        $reason = $context['reason'];

        echo "Transfer for object with identifier: "
            . $progressSnapshot->getIdentifier() . " has failed.";

        echo "Completion Status: " . $progressSnapshot->getTransferredBytes()
            . "/"
            . $progressSnapshot->getTotalBytes() . " B";

        echo "Reason of failure is: " . $reason;
    }
}
```

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use MyApp\Listeners\MyCustomListener;

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

```
// The following example uses `MyCustomListener` in an `upload` operation.
$transferManager = new S3TransferManager(
    null,
    ['default_region' => 'us-east-2']
);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        source: "/tmp/myfile.txt",
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'MyKey'
        ],
        listeners: [
            new MyCustomListener()
        ]
    )
);

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

Pelacakan kemajuan

S3 Transfer Manager menyediakan kemampuan pelacakan kemajuan bawaan.

Melacak semua operasi

Anda dapat mengaktifkan pelacakan untuk semua operasi (file dan direktori) dengan menyetel `track_progress` ke `true` pada `S3TransferManager` instance.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(
    null,
    [
        'track_progress' => true,
    ]
);
```

Dengan pengaturan ini, SDK menggunakan built-in `SingleProgressTracker` *<add link>* untuk operasi file dan menggunakan <add link>operasi direktori `MultiProgressTracker` for.

Melacak operasi file tunggal

Jika Anda tidak mengaktifkan pelacakan kemajuan pada manajer transfer, Anda dapat mengaktifkan pelacakan pada metode individual dengan dua cara.

Setel `track_progress` ke `true` dalam `$config` array untuk operasi file tertentu.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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


$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [   // $config array
            'track_progress' => true,
        ],
        []
    )
);
```

Buat `SingleProgressTracker` instance baru atau berikan implementasi kustom`AbstractTransferListener`. Lewatkan pelacak Anda ke operasi file sebagai `$progressTracker` parameter.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;
```

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/Myfile.txt',  // Or an instance of StreamInterface.
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'file.txt',
        ],
        [],
        [],
        new SingleProgressTracker()  // Or custom implementation of the
 TransferLister interface.
    )
);
```

Contoh berikut menunjukkan contoh keluaran konsol dari bawaan `SingleProgressTracker` untuk kemajuan unggahan file bernama`MyFile.txt`:

```
MyFile.txt:
[##########################                                ] 50% 5242880/10485760 B
```

Melacak operasi direktori tunggal

Jika Anda tidak mengaktifkan pelacakan kemajuan pada manajer transfer, Anda dapat mengaktifkannya untuk operasi direktori dengan dua cara.

Setel 'track_progress' ke `true` dalam argumen array metode. `$config`

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

```
$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [   // $config array
            'track_progress' => true,
        ],
        []
    )
);
```

Buat `MultiProgressTracker` instance baru dan teruskan ke operasi direktori sebagai `$progressTracker` parameter.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\Progress\MultiProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [],
        [],
        new MultiProgressTracker()
    )
);
```

Contoh berikut menunjukkan contoh keluaran konsol dari built-in `MultiProgressTracker` untuk kemajuan upload direktori:

```
MyFile.4.txt:
[##################################################] 100% 4194304/4194304 B
MyFile.5.txt:
[##################################################] 100% 24117248/24117248 B
MyFile.2.txt:
[                                                  ] 0% 0/10485760 B
MyFile.3.txt:
[##################################################] 100% 18874368/18874368 B
MyFile.1.txt:
[##################################################] 100% 1048576/1048576 B
--------------------------------------------------
[########################################          ] 80% Completed: 3/5, Failed: 0/5
```

Kemajuan setiap file terdaftar bersama dengan kemajuan ringkasan di baris terakhir.

Pelacakan kemajuan khusus

Anda dapat menerapkan pelacakan kemajuan Anda sendiri untuk operasi file. Buat kelas yang memperluas `AbstractTransferListener` antarmuka dan meneruskannya ke metode pada permintaan.

Implementasi berikut menampilkan informasi kemajuan dalam format teks untuk operasi file tunggal:

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\AbstractTransferListener;
use Aws\S3\S3Transfer\S3TransferManager;

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

class MyProgressTracker extends AbstractTransferListener
{
    public function transferInitiated(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Starting transfer of {$key}...\n";
    }

    public function bytesTransferred(array $context): bool
    {
        $snapshot = $context['progress_snapshot'];
```

```
        $percent = round($snapshot->ratioTransferred() * 100, 2);
        $transferred = round($snapshot->getTransferredBytes() / 1024 / 1024, 2);
        $total = round($snapshot->getTotalBytes() / 1024 / 1024, 2);

        echo "Progress: {$percent}% ({$transferred}MB of {$total}MB)\n";
    }

    public function transferComplete(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Transfer complete for {$key}!\n";
    }

    public function transferFail(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        $reason = $context['reason']->getMessage();
        echo "Transfer failed for {$key}: {$reason}\n";
    }
}

$transferManager = new S3TransferManager(
    null, []
);

// Use your custom tracker.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt', // Or an instance of StreamInterface.
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ],
        [],
        [],
        new MyProgressTracker()
    )
);
```

Contoh berikut menunjukkan contoh output ke konsol untuk upload file yang sedang berlangsung:

```
Starting transfer of file.txt...
Progress: 42% (16.8MB of 40MB)
```

Sesuaikan pelacak kemajuan bawaan

Anda dapat menyesuaikan pelacakan kemajuan dengan beberapa cara:

- Buat implementasi kustom dengan memperluas `AbstractTransferListener` dan mengimplementasikan `ProgressTrackerInterface`*<add link>*.
- Sesuaikan implementasi yang ada dengan mengonfigurasi `ConsoleProgressBar` *<add link>* yang diteruskan ke pelacak bawaan.
- Menerapkan kustom `ProgressBar` *<add link>* dan meneruskannya ke pelacak bawaan.

Contoh berikut menunjukkan cara menyesuaikan pelacakan kemajuan. Lihat diagram kelas untuk melihat hubungan antara komponen pelacakan kemajuan di SDK.

Kustomisasi **`ConsoleProgressBar`**

Sesuaikan format bilah default

Contoh berikut menyesuaikan format progress bar default, yaitu `ColoredTransferProgressBarFormat`*<add link>*. Format ini menunjukkan kemajuan dengan warna berbeda tergantung pada status transfer.

Contoh berubah:

- karakter dari default “#” ke “=”
- lebar bar dari default 50 hingga 100

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$progressBar = new ConsoleProgressBar(
    progressBarChar: '=',
    progressBarWidth: 100,
// Default `ColoredTransferProgressBarFormat` is used.
```

```
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file.txt";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

Gambar berikut menunjukkan output konsol untuk built-in yang disesuaikan `ConsoleProgressBar` dengan lebar 100 dan karakter "=":

```
key:
[                                                  ] 0% 0/1048576 B

key:
[==================================================] 100% 1048576/1048576 B
```

Sesuaikan dengan menggunakan format bilah kemajuan biasa

Contoh berikut menyesuaikan progress bar default dengan menggunakan built-in `PlainProgressBarFormat` *<add link>* dengan karakter kustom dan lebar. `PlainProgressBarFormat`Tampilan hanya dalam warna hitam. Warnanya tidak berubah berdasarkan status transfer.

Contoh berubah:

- karakter dari default "#" ke "*"
- lebar bar dari default 50 hingga 25

- `progressBarFormat`contoh default dari `ColoredTransferProgressBarFormat` ke `PlainProgressBarFormat`

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\PlainProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$progressBar = new ConsoleProgressBar(
    progressBarChar: '*',
    progressBarWidth: 25,
    progressBarFormat: new PlainProgressBarFormat(),
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file.txt";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

Gambar berikut menunjukkan output konsol untuk built-in yang disesuaikan `ConsoleProgressBar` menggunakan format bilah kemajuan biasa dengan lebar 25 dan karakter “*”:

```
key:
[                              ] 0%

key:
[***********                  ] 44%

key:
[*************************] 100%
```

Sesuaikan dengan membuat format bar baru

Contoh berikut menyesuaikan progress bar default dengan menggunakan format bar baru (extending `ProgressBarFormat`*<add link>*) alih-alih menyesuaikan format bawaan.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\ProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3TransferManager;

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

// Implement a custom bar format.
class PercentBarFormat extends ProgressBarFormat
{
    public const FORMAT_TEMPLATE = "|object_name| - |percent|% [|transferred| |unit|
ytes]";
    public const FORMAT_PARAMETERS = [
        'object_name',       // `uploadRequestArgs` Key parameter.
        'percent',           //  Percent transferred.
        // 'progress_bar',      Optional, if used, default is a `ConsoleProgressBar`.
        // 'to_be_transferred', Optional.
        'transferred',       //  Default is bytes transferred.
        'unit',              //  Default is 'B'.
    ];

    public function getFormatTemplate(): string
    {
```

```
        return self::FORMAT_TEMPLATE;
    }

    public function getFormatParameters(): array
    {
        return self::FORMAT_PARAMETERS;
    }

    protected function getFormatDefaultParameterValues(): array
    {
        return [];
    }
}

// Create an instance of the custom implemention
$progressBarFormat = new PercentBarFormat();
$progressBar = new ConsoleProgressBar(
    progressBarFormat: $progressBarFormat,
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

Gambar berikut menunjukkan output konsol untuk built-in yang disesuaikan `ConsoleProgressBar` menggunakan format bilah kemajuan khusus:

```
key - 0% [0 Bytes]
key - 11% [1153434 Bytes]
key - 55% [5767168 Bytes]
key - 100% [10485760 Bytes]
```

Format bilah khusus menerima beberapa parameter:

- object_name
- persen
- progress_bar
- to_be_ditransfer
- ditransfer
- unit

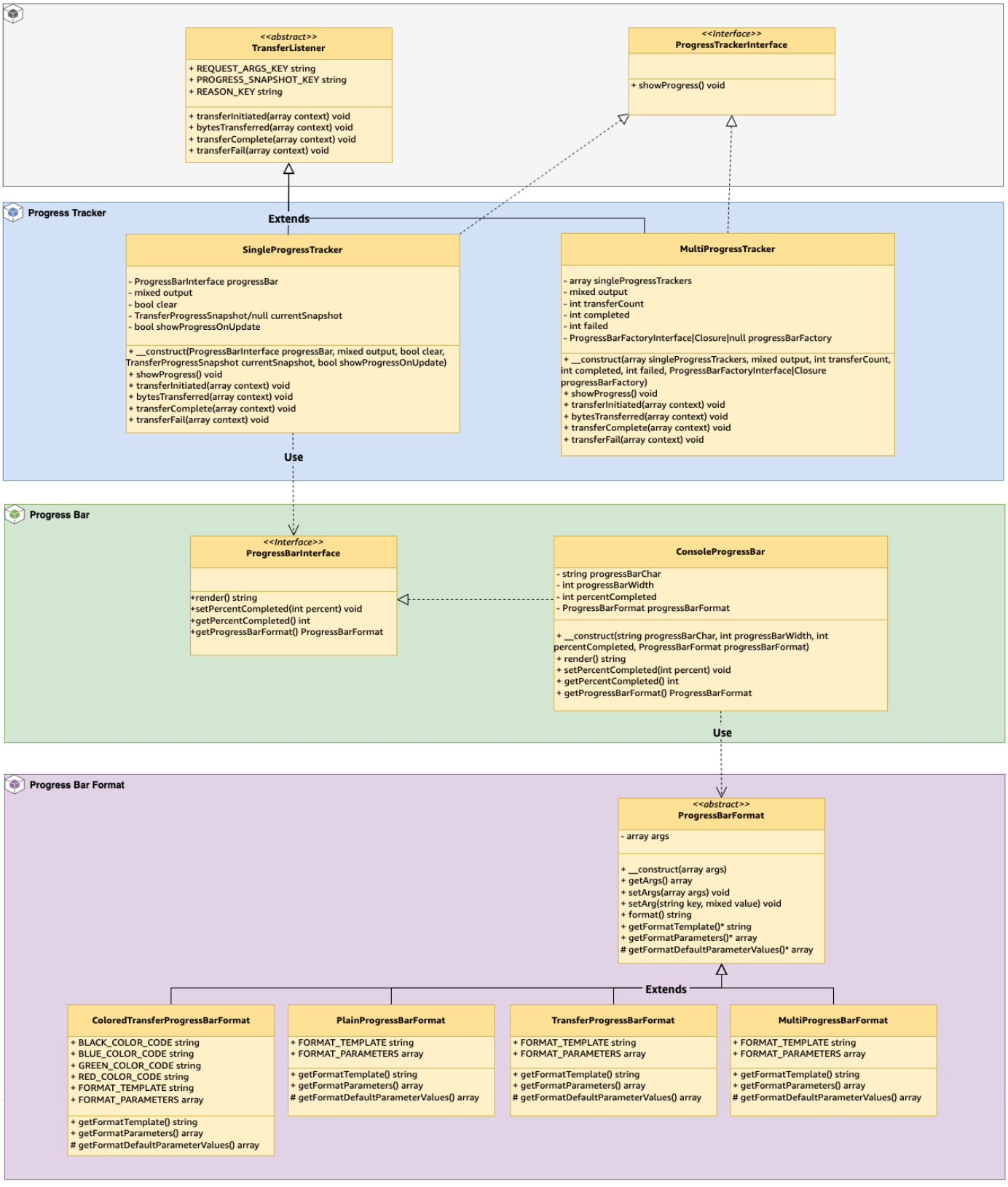
Anda dapat menggabungkan nama parameter (karakter pipa) yang dibatasi untuk menyesuaikan output. Contoh menggunakan template berikut. `getFormatTemplate`Metode `PercentBarFormat` kelas mengembalikan template ini:

```
"|object_name| - |percent|% [|transferred| |unit|ytes]"
```

Diagram kelas: Komponen pelacakan Manajer Transfer S3

Diagram kelas berikut menunjukkan bagaimana komponen pelacakan bekerja sama di S3 Transfer Manager. Memahami hubungan ini membantu Anda menerapkan pendengar khusus dan pelacakan kemajuan.

S3 Transfer Manager tracking components



Penanganan kesalahan

S3 Transfer Manager menggunakan janji untuk operasi asinkron. Anda dapat menangani kesalahan menggunakan janji `otherwise` atau `then` metode. Anda juga dapat membungkus implementasi Anda dalam satu `try-catch` blok.

Penanganan kesalahan janji

Menggunakan`otherwise` :

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);

$uploadPromise->otherwise(function (Throwable $reason) {
    echo "Upload failed: " . $reason->getMessage() . "\n";
});
```

Menggunakan`then` :

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Models\UploadResult;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
```

```
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);


$uploadPromise->then(
    function (UploadResult $result) {
        echo "Upload succeeded!\n";
    },
    function (Throwable $error) {
        echo "Upload failed: " . $error->getMessage() . "\n";
    }
)->wait();
```

Menggunakan blok try-catch

```
<?php

use Aws\S3\S3Transfer\Exception\S3TransferException;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'file.txt',
        ]
    )
);

try {
```

```
    $uploadPromise->wait();
} catch (S3TransferException $exception) {
    echo "Upload failed: " . $exception->getMessage() . "\n";
}
```

<Add more details about S3TransferManager exception and also a link to the API reference for it.>

Kebijakan kegagalan operasi direktori

Untuk operasi direktori, Anda juga dapat menentukan callback kebijakan kegagalan. Callback ini menangani kegagalan untuk file individual dan memutuskan apakah pengunggahan direktori harus dilanjutkan.

Example menggunakan fungsi 'failure_policy' untuk operasi **`uploadDirectory`**

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $uploadDirectoryRequestArgs,
                $reason,
                $uploadDirectoryResponse
            ) {
                echo "Failed to upload {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}\n";
                echo "So far, uploaded: " .
                    "{$uploadDirectoryResponse->getObjectsUploaded()}, " .
                    "failed: {$uploadDirectoryResponse->getObjectsFailed()}\n";
```

```
                // Return true to continue with other files,
                // or throw an exception to abort.
                return true;
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Example menggunakan fungsi 'failure_policy' untuk operasi **`downloadDirectory`**

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Log errors but continue with other files.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/directory',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $downloadDirectoryRequestArgs,
                $reason,
                $downloadDirectoryResponse
            ) {
                error_log("Failed to download {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}");

                // If we've had too many failures, abort the entire operation.
                if ($downloadDirectoryResponse->getObjectsFailed() > 10) {
                    throw new \Exception(
                        "Too many download failures, aborting operation"
                    );
                }
```

```
                // Return void to continue with other files.
                return;
            },
        ]
    )
);
$downloadDirPromise->wait();
```

Penggunaan lanjutan

Bagian ini mencakup pola dan teknik penggunaan lanjutan untuk S3 Transfer Manager.

Unggahan multipart

S3 Transfer Manager secara otomatis menggunakan unggahan multipart untuk file besar. Anda dapat menyesuaikan perilaku ini dengan opsi konfigurasi.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/large/file.mp4',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'    => 'videos/large-file.mp4',
        ],
        [
            // Use multipart upload for files larger than 100MB.
            'multipart_upload_threshold_bytes' => 100 * 1024 * 1024,

            // Use 25MB parts for multipart uploads.
            'target_part_size_bytes'           => 25 * 1024 * 1024,
        ]
    )
);
$uploadPromise->wait();
```

Unduhan multibagian

Anda dapat menyesuaikan perilaku unduhan multibagian.

```
<?php

use Aws\S3\S3Transfer\AbstractMultipartDownloader;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$downloadPromise = $transferManager->download(
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/large-file.mp4',
        [],
        [
            // Use 25MB parts for multipart downloads.
            'target_part_size_bytes'  => 25 * 1024 * 1024,

            // Use ranged-based download instead of part-based.
            'multipart_download_type' =>
 AbstractMultipartDownloader::RANGED_GET_MULTIPART_DOWNLOADER,
        ]
    )
);
$downloadPromise->wait();
```

Untuk informasi selengkapnya tentang anggota `AbstractMultipartDownloader` kelas lainnya, lihat dokumentasi API*<add link>*.

Filter khusus

Anda dapat menggunakan fungsi filter untuk mengunggah atau mengunduh file secara selektif untuk operasi direktori.

Unggah filter direktori

Untuk informasi selengkapnya tentang informasi parameter, lihat opsi filter callable dari `uploadDirectory` metode ini.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Upload files modified in the last 24 hours.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'filter' => function ($file) {
                $modTime = filemtime($file);
                return (time() - $modTime) < 86400; // 24 hours
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Unduh filter direktori

Untuk informasi selengkapnya tentang informasi parameter, lihat opsi filter callable dari `downloadDirectory` metode ini.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Download files with a specific prefix.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
```

```
        '/path/to/directory',
        [],
        [
            'filter' => function ($key) {
                return strpos($key, 'reports/2023/') === 0;
            },
        ]
    )
);
$downloadDirPromise->wait();
```

Panggilan balik pra-permintaan

Anda dapat mengubah parameter permintaan untuk setiap file menggunakan callback pra-permintaan. Untuk informasi selengkapnya tentang parameter, lihat berikut ini:

- put_object_request_callaback opsi callable dari metode `uploadDirectory`
- download_object_request_modifier opsi callable dari metode `downloadDirectory`

Example dari panggilan balik pra-permintaan untuk metode **`uploadDirectory`**

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Set custom metadata for each uploaded file.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'upload_object_request_modifier' => function ($args) {
                $extension = pathinfo($args['Key'], PATHINFO_EXTENSION);

                switch ($extension) {
                    case 'jpg':
```

```
                    case 'jpeg':
                        $args['ContentType'] = 'image/jpeg';
                        break;
                    case 'png':
                        $args['ContentType'] = 'image/png';
                        break;
                    case 'pdf':
                        $args['ContentType'] = 'application/pdf';
                        break;
                }

                $args['Metadata'] = [
                    'uploaded_at' => date('Y-m-d H:i:s'),
                ];
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Ready-to-use contoh

Bagian ini menyediakan kelas pembantu lengkap dengan metode statis untuk operasi S3 Transfer Manager umum. Kelas pembantu menyederhanakan penggunaan. Ini menangani inisialisasi dan konfigurasi secara internal.

Kelas S3 TransferHelper

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use Psr\Http\Message\StreamInterface;

class S3TransferHelper
{
    /**
     * Upload a file from a local path to S3.
     */
```

```
    public static function uploadFile(
        string $filePath,
        string $bucket,
        string $key
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $filePath,
            [
                'Bucket' => $bucket,
                'Key'    => $key,
            ],
            []
        );

        return $transferManager->upload($uploadRequest)->wait()->getResult();
    }

    /**
     * Upload from a stream to S3
     */
    public static function uploadFromStream(
        StreamInterface $stream,
        string          $bucket,
        string          $key,
        bool            $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $stream,
            [
                'Bucket' => $bucket,
                'Key'    => $key,
            ],
            ['track_progress' => $trackProgress]
        );

        return $transferManager->upload($uploadRequest)->wait()->getResult();
    }
```

```
    /**
     * Download a file from S3 to a local path
     */
    public static function downloadFile(
        string $bucket,
        string $key,
        string $destinationPath,
        bool   $failsWhenDestinationExists = false,
        bool   $trackProgress = true,
    ): void
    {
        $transferManager = new S3TransferManager();
        $downloadRequest = new DownloadRequest(
            [
                'Bucket' => $bucket,
                'Key'    => $key
            ],
            [],
            ['track_progress' => $trackProgress]
        );

        $downloadFileRequest = new DownloadFileRequest(
            $destinationPath,
            $failsWhenDestinationExists,
            $downloadRequest
        );

        $transferManager->downloadFile($downloadFileRequest)->wait();
    }

    /**
     * Upload an entire directory to S3
     */
    public static function uploadDirectory(
        string $sourceDirectory,
        string $bucket,
        string $s3Prefix = '',
        bool   $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();
        $uploadDirectoryRequest = new UploadDirectoryRequest(
            $sourceDirectory,
            $bucket,
```

```
            [],
            [
                's3_prefix'      => $s3Prefix,
                'track_progress' => $trackProgress,
            ]
        );

        $result = $transferManager->uploadDirectory($uploadDirectoryRequest)->wait();

        return [
            'uploaded' => $result->getObjectsUploaded(),
            'failed'   => $result->getObjectsFailed(),
        ];
    }

    /**
     * Download directory from S3
     */
    public static function downloadDirectory(
        string $bucket,
        string $destinationDirectory,
        string $s3Prefix = '',
        bool   $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();
        $downloadDirectoryRequest = new DownloadDirectoryRequest(
            $bucket,
            $destinationDirectory,
            [],
            [
                's3_prefix'      => $s3Prefix,
                'track_progress' => $trackProgress,
            ]
        );

        $result = $transferManager->downloadDirectory($downloadDirectoryRequest)-
>wait();

        return [
            'downloaded' => $result->getObjectsDownloaded(),
            'failed'     => $result->getObjectsFailed(),
        ];
    }
```

```
}
```

Contoh penggunaan kelas pembantu

Contoh berikut menunjukkan cara menggunakan `S3TransferHelper` kelas.

```
<?php

require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/S3TransferHelper.php';

// Upload a local file
$result = S3TransferHelper::uploadFile(
    '/path/to/local/document.pdf',
    'amzn-s3-demo-bucket',
    'documents/document.pdf'
);

echo "File uploaded successfully. ETag: " . $result['ETag'] . "\n";

// Download a file to local path
S3TransferHelper::downloadFile(
    'amzn-s3-demo-bucket',
    'documents/document.pdf',
    '/path/to/local/downloaded-document.pdf',
    false, // Don't fail if destination exists
    true   // Track progress
);

echo "File downloaded successfully!\n";

// Upload entire directory
$result = S3TransferHelper::uploadDirectory(
    '/path/to/local/photos',
    'amzn-s3-demo-bucket',
    'photos/vacation-2023/', // S3 prefix
    true // Track progress
);

echo "Directory upload completed.\n";
echo "Uploaded: {$result['uploaded']} files\n";
echo "Failed: {$result['failed']} files\n";

// Download directory from S3
```

```
$result = S3TransferHelper::downloadDirectory(
    'amzn-s3-demo-bucket',
    '/path/to/local/downloads',
    'photos/vacation-2023/', // S3 prefix to download
    true // Track progress
);

echo "Directory download completed.\n";
echo "Downloaded: {$result['downloaded']} files\n";
echo "Failed: {$result['failed']} files\n";
```

Transfer direktori ke dan dari Amazon S3 menggunakan AWS SDK for PHP Versi 3

Anda menggunakan `Transfer` kelas di AWS SDK for PHP Versi 3 untuk mengunggah seluruh direktori ke bucket Amazon S3 dan mengunduh seluruh bucket ke direktori lokal.

Mengunggah direktori lokal ke Amazon S3

`Aws\S3\Transfer`Objek melakukan transfer. Contoh berikut menunjukkan cara mengunggah direktori file lokal secara rekursif ke bucket Amazon S3.

```
// Create an S3 client.
$client = new \Aws\S3\S3Client([
    'region'  => 'us-west-2',
    'version' => '2006-03-01',
]);

// Where the files will be sourced from.
$source = '/path/to/source/files';

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

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

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

Dalam contoh ini, kami membuat klien Amazon S3, membuat `Transfer` objek, dan melakukan transfer secara sinkron.

Contoh sebelumnya menunjukkan jumlah minimum kode yang diperlukan untuk melakukan transfer. Sebuah `Transfer` objek juga dapat melakukan transfer secara asinkron dan memiliki berbagai opsi konfigurasi yang dapat Anda gunakan untuk menyesuaikan transfer.

Anda dapat mengunggah file lokal ke “subfolder” bucket Amazon S3 dengan menyediakan key prefix di URI. `s3://` Contoh berikut mengunggah file lokal pada disk ke `bucket` bucket dan menyimpan file di bawah `foo` key prefix.

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
$directoryTransfer->transfer();
```

Mengunduh bucket Amazon S3

Anda dapat mengunduh bucket Amazon S3 secara rekursif ke direktori lokal pada disk dengan menentukan `$source` argumen sebagai URI Amazon S3 (mis.`s3://bucket`,) dan argumen sebagai jalur `$dest` ke direktori lokal.

```
// Where the files will be sourced from.
$source = 's3://bucket';

// Where the files will be transferred to.
$dest = '/path/to/destination/dir';

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

> **Note**
>
> SDK secara otomatis membuat direktori yang diperlukan saat mengunduh objek di ember.

Anda dapat menyertakan key prefix di Amazon S3 URI setelah bucket hanya mengunduh objek yang disimpan di bawah “pseudo-folder”. Contoh berikut hanya mengunduh file yang disimpan di bawah awalan kunci “/foo” dari bucket yang diberikan.

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
```

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

Konfigurasi

Konstruktor `Transfer` objek menerima argumen berikut.

`$client`

`Aws\ClientInterface`Objek yang digunakan untuk melakukan transfer.

`$source`(string |**`Iterator`**)

Sumber data yang sedang ditransfer. Ini dapat menunjuk ke jalur lokal pada disk (misalnya,`/path/to/files`) atau bucket Amazon S3 (mis.,`s3://bucket`). `s3://`URI juga dapat berisi key prefix yang dapat digunakan untuk hanya mentransfer objek di bawah awalan umum.

Jika `$source` argumennya adalah URI Amazon S3, `$dest` argumennya harus berupa direktori lokal (dan sebaliknya).

Selain memberikan nilai string, Anda juga dapat memberikan `\Iterator` objek yang menghasilkan nama file absolut. Jika Anda memberikan `\Iterator` objek, Anda harus memberikan `base_dir` opsi dalam array `$options` asosiatif.

`$dest`

Tujuan di mana file akan ditransfer. Jika `$source` argumennya adalah jalur lokal pada disk, `$dest` harus berupa URI bucket Amazon S3 (misalnya,`s3://bucket`). Jika `$source` argumennya adalah URI bucket Amazon S3, `$dest` argumennya harus berupa jalur lokal pada disk.

`$options`

Array asosiatif opsi transfer. Opsi transfer berikut valid:

`add_content_md5`(bool)

Setel `true` untuk menghitung MD5 checksum untuk upload.

`base_dir`(tali)

Direktori dasar sumber, jika `$source` adalah iterator. Jika `$source` opsi ini bukan array, maka opsi ini diabaikan.

`before`(dapat dipanggil)

Panggilan balik untuk dipanggil sebelum setiap transfer. Callback harus memiliki tanda tangan fungsi seperti`function (Aws\Command $command) {...}`. Perintah yang diberikan akan berupa`GetObject`,`PutObject`,`CreateMultipartUpload`,`UploadPart`, atau `CompleteMultipartUpload` perintah.

`mup_threshold`(int)

Ukuran dalam byte di mana unggahan multibagian harus digunakan sebagai pengganti. `PutObject` Default ke `16777216` (16 MB).

`concurrency`(int, default=5)

Jumlah file yang akan diunggah secara bersamaan. Nilai konkurensi yang ideal akan bervariasi berdasarkan jumlah file yang diunggah dan ukuran rata-rata setiap file. Umumnya, file yang lebih kecil mendapat manfaat dari konkurensi yang lebih tinggi sementara file yang lebih besar tidak.

`debug`(bool)

Setel `true` untuk mencetak informasi debug untuk transfer. Setel ke `fopen()` sumber daya untuk menulis ke aliran tertentu alih-alih menulis ke STDOUT.

Transfer async

`Transfer`Objek adalah contoh dari`GuzzleHttp\Promise\PromisorInterface`. Ini berarti bahwa transfer dapat terjadi secara asinkron dan dimulai dengan memanggil `promise` metode objek.

```
$source = '/path/to/source/files';
$dest = 's3://bucket';
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);

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

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

Janji ditolak jika ada file yang gagal ditransfer. Anda dapat menangani transfer yang gagal secara asinkron menggunakan `otherwise` metode janji. `otherwise`Fungsi menerima callback untuk

memanggil ketika terjadi kesalahan. Callback menerima penolakan, yang biasanya merupakan instance dari `Aws\Exception\AwsException` (meskipun nilai jenis apa pun dapat dikirimkan ke callback). `$reason`

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

Karena `Transfer` objek mengembalikan janji, transfer ini dapat terjadi bersamaan dengan janji asinkron lainnya.

Menyesuaikan transfer direktori

Dengan menambahkan callback ke konstruktor, Anda dapat menyesuaikan opsi yang dijalankan. `Transfer`

```
$uploader = new Transfer($s3Client, $source, $dest, [
    'before' => function (\Aws\Command $command) {
        // Commands can vary for multipart uploads, so check which command
        // is being processed.
        if (in_array($command->getName(), ['PutObject', 'CreateMultipartUpload'])) {
            // Set custom cache-control metadata.
            $command['CacheControl'] = 'max-age=3600';
            // Apply a canned ACL.
            $command['ACL'] = strpos($command['Key'], 'CONFIDENTIAL') ### false
                ? 'public-read'
                : 'private';
        }
    },
]);
```

Enkripsi sisi klien Amazon S3 di Versi 3 AWS SDK for PHP

Dengan enkripsi sisi klien, data dienkripsi dan didekripsi langsung di lingkungan Anda. Ini berarti bahwa data ini dienkripsi sebelum ditransfer ke Amazon S3, dan Anda tidak bergantung pada layanan eksternal untuk menangani enkripsi untuk Anda. Untuk implementasi baru, kami menyarankan penggunaan `S3EncryptionClientV3` dan `S3EncryptionMultipartUploaderV3` over the `S3EncryptionClientV2` and and `S3EncryptionClient` deprecated `S3EncryptionMultipartUploaderV2` and. `S3EncryptionMultipartUploader` Disarankan agar implementasi yang lebih lama masih menggunakan versi yang tidak digunakan lagi mencoba

untuk bermigrasi. `S3EncryptionClientV3`mempertahankan dukungan untuk mendekripsi data yang dienkripsi menggunakan warisan. `S3EncryptionClient`

AWS SDK for PHP Mengimplementasikan enkripsi amplop dan menggunakan OpenSSL untuk mengenkripsi dan mendekripsi. Implementasinya dapat dioperasikan dengan yang lain SDKs yang sesuai dengan dukungan fiturnya. Ini juga kompatibel dengan alur kerja asinkron berbasis janji SDK.

Panduan migrasi

Bagi mereka yang mencoba bermigrasi dari klien yang tidak digunakan lagi ke klien baru, ada panduan migrasi untuk bermigrasi dari v1 ke v2 di sini dan panduan migrasi untuk beralih dari v2 ke v3 di sini.

Pengaturan

Untuk memulai enkripsi sisi klien, Anda memerlukan yang berikut ini:

- Kunci AWS KMS enkripsi
- Ember S3

Sebelum menjalankan kode contoh apa pun, konfigurasikan AWS kredensil Anda. Lihat Kredensial untuk AWS SDK for PHP Versi 3.

Enkripsi

Mengunggah objek terenkripsi `S3EncryptionClientV3` membutuhkan empat parameter tambahan di atas parameter standar: `PutObject`

- `'@KmsEncryptionContext'`adalah pasangan kunci-nilai yang dapat digunakan untuk menambahkan lapisan keamanan ekstra ke objek terenkripsi Anda. Klien enkripsi harus meneruskan kunci yang sama, yang secara otomatis akan dilakukan pada panggilan get. Jika tidak ada konteks tambahan yang diinginkan, teruskan dalam array kosong.
- `@CipherOptions`adalah konfigurasi tambahan untuk enkripsi termasuk sandi mana yang akan digunakan dan ukuran kunci.
- `@MaterialsProvider`adalah penyedia yang menangani pembuatan kunci sandi dan vektor inisialisasi, serta mengenkripsi kunci sandi Anda.
- `@CommitmentPolicy`adalah opsi kebijakan yang menentukan bagaimana suatu objek dibaca baik dengan komitmen kunci atau tanpa komitmen utama dan bagaimana suatu objek ditulis dengan komitmen utama atau tanpa komitmen utama.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV3;

 // Let's construct our S3EncryptionClient using an S3Client
 $encryptionClient = new S3EncryptionClientV3(
     new S3Client([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ])
 );

 $kmsKeyId = 'kms-key-id';
 $materialsProvider = new KmsMaterialsProviderV3(
     new KmsClient([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ]),
     $kmsKeyId
 );

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

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

> **Note**
>
> Selain Amazon S3 dan kesalahan layanan AWS KMS berbasis, Anda mungkin menerima `InvalidArgumentException` objek yang dilemparkan jika Anda tidak `'@CipherOptions'` dikonfigurasi dengan benar.

Dekripsi

Mengunduh dan mendekripsi objek memiliki lima parameter tambahan, dua di antaranya diperlukan, di atas parameter standar`GetObject`. Klien akan mendeteksi opsi sandi dasar untuk Anda.

- **`'@SecurityProfile'`**: Jika disetel ke 'V3', hanya objek yang dienkripsi dalam V3-kompatibel

  Format dapat didekripsi. Menyetel parameter ini ke 'V3_AND_LEGACY' juga memungkinkan objek yang dienkripsi dalam format yang kompatibel dengan V1 untuk didekripsi. Untuk mendukung migrasi, setel @ ke SecurityProfile 'V3_AND_LEGACY'. Gunakan 'V3' hanya untuk pengembangan aplikasi baru.
- **`'@MaterialsProvider'`**adalah penyedia yang menangani pembuatan kunci sandi dan vektor inisialisasi, sebagai

  serta mengenkripsi kunci cipher Anda.
- **`'@KmsAllowDecryptWithAnyCmk'`**: (opsional) Menyetel parameter ini ke true memungkinkan dekripsi

  tanpa memasok id kunci KMS ke konstruktor file. MaterialsProvider Nilai default adalah false.
- **`'@CipherOptions'`**(opsional) adalah konfigurasi tambahan untuk enkripsi termasuk

  cipher untuk digunakan dan keysize.
- **`@CommitmentPolicy`**opsi kebijakan yang menentukan bagaimana suatu objek dibaca

  komitmen kunci atau tanpa komitmen utama dan bagaimana suatu objek ditulis dengan komitmen utama atau tanpa komitmen utama.

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
```

```
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> Selain Amazon S3 dan kesalahan layanan AWS KMS berbasis, Anda mungkin menerima `InvalidArgumentException` objek yang dilemparkan jika Anda tidak `'@CipherOptions'` dikonfigurasi dengan benar.

Konfigurasi cipher

`'Cipher'` (tali)

Metode cipher yang digunakan klien enkripsi saat mengenkripsi. Hanya 'gcm' yang didukung saat ini.

> **Important**
>
> PHP diperbarui dalam versi 7.1 untuk menyertakan parameter tambahan yang diperlukan untuk mengenkripsi dan mendekripsi menggunakan OpenSSL untuk enkripsi GCM. Untuk PHP versi 7.0 dan sebelumnya, polyfill untuk dukungan GCM disediakan dan digunakan oleh klien enkripsi dan. `S3EncryptionClientV2` `S3EncryptionMultipartUploaderV2` Namun, kinerja untuk input besar akan jauh lebih lambat menggunakan polyfill daripada menggunakan implementasi asli untuk PHP 7.1+, sehingga meningkatkan lingkungan versi PHP yang lebih lama mungkin diperlukan untuk menggunakannya secara efektif.

`'KeySize'` (int)

Panjang kunci enkripsi konten yang akan dihasilkan untuk mengenkripsi. Default ke 256 bit. Opsi konfigurasi yang valid adalah 256 bit.

`'Aad'` (tali)

Opsional 'Data otentikasi tambahan' untuk disertakan dengan muatan terenkripsi Anda. Informasi ini divalidasi pada dekripsi. `Aad`hanya tersedia saat menggunakan cipher 'gcm'.

> **Important**
>
> Data otentikasi tambahan tidak didukung oleh semua AWS SDKs dan karena itu orang lain SDKs mungkin tidak dapat mendekripsi file yang dienkripsi menggunakan parameter ini.

Strategi metadata

Anda juga memiliki pilihan untuk menyediakan sebuah instance dari kelas yang mengimplementasikan. `Aws\Crypto\MetadataStrategyInterface` Antarmuka sederhana ini menangani penyimpanan dan pemuatan `Aws\Crypto\MetadataEnvelope` yang berisi materi enkripsi amplop Anda. SDK menyediakan dua kelas yang mengimplementasikan ini: `Aws\S3\Crypto\HeadersMetadataStrategy` dan`Aws\S3\Crypto\InstructionFileMetadataStrategy`. `HeadersMetadataStrategy`digunakan secara default.

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

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

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V3',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Konstanta nama kelas untuk `HeadersMetadataStrategy` dan juga `InstructionFileMetadataStrategy` dapat diberikan dengan memanggil: :class.

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

> **Note**
>
> Jika ada kegagalan setelah file instruksi diunggah, itu tidak akan dihapus secara otomatis.

Unggahan multipart

Melakukan unggahan multibagian dengan enkripsi sisi klien juga dimungkinkan. `Aws\S3\Crypto\S3EncryptionMultipartUploaderV3`Mempersiapkan aliran sumber untuk enkripsi sebelum mengunggah. Membuat satu mengambil pengalaman yang sama dengan menggunakan `Aws\S3\MultipartUploader` dan`Aws\S3\Crypto\S3EncryptionClientV3`. `S3EncryptionMultipartUploaderV3`Dapat menangani `'@MetadataStrategy'` opsi yang sama seperti`S3EncryptionClientV3`, serta semua `'@CipherOptions'` konfigurasi yang tersedia.

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-upload-key';
$cipherOptions = [
```

```
    'Cipher' => 'gcm'
    'KeySize' => 256,
    // Additional configuration options
];

$multipartUploader = new S3EncryptionMultipartUploaderV3(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
    [
        '@MaterialsProvider' => $materialsProvider,
        '@CipherOptions' => $cipherOptions,
        '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
        'bucket' => $bucket,
        'key' => $key,
    ]
);
$multipartUploader->upload();
```

> **Note**
>
> Selain Amazon S3 dan kesalahan layanan AWS KMS berbasis, Anda mungkin menerima `InvalidArgumentException` objek yang dilemparkan jika Anda tidak `'@CipherOptions'` dikonfigurasi dengan benar.

Perlindungan integritas data dengan checksum

Amazon Simple Storage Service (Amazon S3) menyediakan kemampuan untuk menentukan checksum saat Anda mengunggah objek. Ketika Anda menentukan checksum, itu disimpan dengan objek dan dapat divalidasi ketika objek diunduh.

Checksum menyediakan lapisan integritas data tambahan saat Anda mentransfer file. Dengan checksum, Anda dapat memverifikasi konsistensi data dengan mengonfirmasi bahwa file yang diterima cocok dengan file asli. Untuk informasi selengkapnya tentang checksum dengan Amazon S3, lihat Panduan Pengguna Layanan Penyimpanan Sederhana Amazon termasuk algoritme yang didukung.

Anda memiliki fleksibilitas untuk memilih algoritma yang paling sesuai dengan kebutuhan Anda dan membiarkan SDK menghitung checksum. Atau, Anda dapat memberikan nilai checksum yang telah dihitung sebelumnya dengan menggunakan salah satu algoritme yang didukung.

> **Note**
>
> Dimulai dengan versi 3.337.0 AWS SDK for PHP, SDK menyediakan perlindungan integritas default dengan secara otomatis menghitung checksum untuk `CRC32` upload. SDK menghitung checksum ini jika Anda tidak memberikan nilai checksum yang telah dihitung sebelumnya atau jika Anda tidak menentukan algoritme yang harus digunakan SDK untuk menghitung checksum.
> SDK juga menyediakan pengaturan global untuk perlindungan integritas data yang dapat Anda atur secara eksternal, yang dapat Anda baca di Panduan Referensi Alat AWS SDKs dan Alat.

> **Important**
>
> Untuk bekerja dengan `CRC32C` algoritma, lingkungan PHP Anda memerlukan instalasi ekstensi AWS Common Runtime (AWS CRT).

Kami membahas checksum dalam dua fase permintaan: mengunggah objek dan mengunduh objek.

Mengunggah objek

Anda meng-upload objek ke Amazon S3 dengan menggunakan metode putObject dari file. `S3Client` Gunakan `ChecksumAlgorithm` pasangan dalam array parameter untuk mengaktifkan perhitungan checksum dan menentukan algoritma.

```
$client = new \Aws\S3\S3Client(['region' => 'us-east-2']); // See the note below.
$result = $client->putObject([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'key',
        'ChecksumAlgorithm' => 'CRC32',
        'Body' => 'Object contents to test the checksum.'
]);
```

Jika Anda tidak menyediakan algoritma checksum dengan permintaan, perilaku checksum bervariasi tergantung pada versi SDK yang Anda gunakan seperti yang ditunjukkan pada tabel berikut.

Perilaku checksum ketika tidak ada algoritma checksum yang disediakan

Versi PHP SDK	Perilaku checksum
lebih awal dari 3.337.0	SDK tidak secara otomatis menghitung checksum berbasis CRC dan menyediakannya dalam permintaan.
3.337.0 atau yang lebih baru	SDK menggunakan `CRC32` algoritma untuk menghitung checksum dan menyediakannya dalam permintaan. Amazon S3 memvalida si integritas transfer dengan menghitung checksumnya sendiri dan membandingkannya dengan `CRC32` checksum yang disediakan oleh SDK. Jika checksum cocok, checksum disimpan dengan objek.

Gunakan nilai checksum yang telah dihitung sebelumnya

Nilai checksum yang telah dihitung sebelumnya yang disertakan dengan permintaan menonaktifkan komputasi otomatis oleh SDK dan menggunakan nilai yang disediakan sebagai gantinya.

Contoh berikut menunjukkan permintaan dengan SHA256 checksum yang telah dihitung sebelumnya.

```
use Aws\S3\S3Client;
use GuzzleHttp\Psr7;

$client = new S3Client([
    'region' => 'us-east-1',
]);

// Calculate the SHA256 checksum of the contents to be uploaded.
$contents = 'Object contents to test the checksum.';
$body = Psr7\Utils::streamFor($contents);
$sha256 = base64_encode(Psr7\Utils::hash($body, 'sha256', true));
```

```
$result = $client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'key',
    'Body' => $body,
    'ChecksumSHA256' => $sha256
]);
```

Jika Amazon S3 menentukan nilai checksum salah untuk algoritme yang ditentukan, layanan akan mengembalikan respons kesalahan.

Unggahan multipart

Anda juga dapat menggunakan checksum dengan unggahan multipart.

Seperti yang ditunjukkan pada contoh berikut, tentukan algoritma checksum sebagai pasangan kunci-nilai dalam `params` array konstruktor. `MultipartUploader`

```
$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$stream = fopen("/path/to/large/file", "r");

$mpUploader = new MultipartUploader($s3Client, $stream, [
    'bucket' => 'amzn-s3-demo-bucket',
    'key' => 'key',
    'params' => ['ChecksumAlgorithm' => 'CRC32']
]);
```

Unduh objek

Saat Anda menggunakan metode getObject untuk mengunduh objek, SDK secara otomatis memvalidasi checksum saat nilai kuncinya. `ChecksumMode` `enabled`

Permintaan dalam cuplikan berikut mengarahkan SDK untuk memvalidasi checksum dalam respons dengan menghitung checksum dan membandingkan nilainya.

```
$result = $client->getObject([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key' => 'test-checksum-key',
        'ChecksumMode' => 'enabled',
]);
```

> **Note**
>
> Jika objek tidak diunggah dengan checksum, tidak ada validasi yang terjadi.

Contoh kode dengan panduan untuk AWS SDK for PHP

Bagian ini berisi contoh kode yang menunjukkan AWS skenario umum yang menggunakan AWS SDK for PHP.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Topik

- CloudFront Contoh Amazon menggunakan AWS SDK for PHP Versi 3
- Menandatangani permintaan CloudSearch domain Amazon khusus dengan AWS SDK for PHP Versi 3
- CloudWatch Contoh Amazon menggunakan AWS SDK for PHP Versi 3
- EC2 Contoh Amazon menggunakan AWS SDK for PHP Versi 3
- Menandatangani permintaan pencarian OpenSearch Layanan Amazon dengan AWS SDK for PHP Versi 3
- AWS Identity and Access Management contoh menggunakan AWS SDK for PHP Versi 3
- AWS Key Management Service contoh menggunakan AWS SDK for PHP Versi 3
- Contoh Amazon Kinesis menggunakan Versi 3 AWS SDK for PHP
- AWS Elemental MediaConvert contoh menggunakan AWS SDK for PHP Versi 3
- Contoh Amazon S3 menggunakan Versi 3 AWS SDK for PHP
- Mengelola rahasia menggunakan Secrets Manager API dan AWS SDK for PHP Versi 3
- Contoh Amazon SES menggunakan AWS SDK for PHP Versi 3
- Contoh Amazon SNS menggunakan Versi 3 AWS SDK for PHP
- Contoh Amazon SQS menggunakan Versi 3 AWS SDK for PHP

- Kirim acara ke titik akhir EventBridge global Amazon menggunakan AWS SDK for PHP Versi 3

CloudFront Contoh Amazon menggunakan AWS SDK for PHP Versi 3

Amazon CloudFront adalah layanan AWS web yang mempercepat penyajian konten web statis dan dinamis dari server web atau server Anda sendiri, seperti Amazon S3. AWS CloudFront mengirimkan konten melalui jaringan pusat data di seluruh dunia yang disebut lokasi tepi. Saat pengguna meminta konten yang Anda distribusikan CloudFront, mereka diarahkan ke lokasi tepi yang memberikan latensi terendah. Jika konten belum di-cache di sana, CloudFront mengambil salinan dari server asal, menyajikannya, dan kemudian men-cache untuk permintaan future.

Untuk informasi selengkapnya CloudFront, lihat Panduan CloudFront Pengembang Amazon.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Mengelola CloudFront distribusi Amazon menggunakan CloudFront API dan AWS SDK for PHP Versi 3

Amazon CloudFront menyimpan konten di lokasi edge di seluruh dunia untuk mempercepat distribusi file statis dan dinamis yang Anda simpan di server Anda sendiri, atau di layanan Amazon seperti Amazon S3 dan Amazon. EC2 Saat pengguna meminta konten dari situs web Anda, CloudFront layani dari lokasi tepi terdekat, jika file tersebut di-cache di sana. Jika tidak, CloudFront mengambil salinan file, menyajikannya, dan kemudian menyimpannya untuk permintaan berikutnya. Caching konten di lokasi tepi mengurangi latensi permintaan pengguna serupa di area tersebut.

Untuk setiap CloudFront distribusi yang Anda buat, Anda menentukan lokasi konten dan cara mendistribusikannya saat pengguna membuat permintaan. Topik ini berfokus pada distribusi untuk file statis dan dinamis seperti HTML, CSS, JSON, dan file gambar. F atau informasi tentang penggunaan CloudFront dengan video sesuai permintaan, lihat On-Demand dan Live Streaming Video dengan CloudFront.

Contoh berikut menunjukkan cara:

- Buat distribusi menggunakan CreateDistribution.
- Dapatkan distribusi menggunakan GetDistribution.
- Daftar distribusi menggunakan ListDistributions.
- Perbarui distribusi menggunakan UpdateDistributions.
- Nonaktifkan distribusi menggunakan DisableDistribution.

- Hapus distribusi menggunakan DeleteDistributions.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensional Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan Amazon CloudFront, lihat Panduan CloudFront Pengembang Amazon.

Buat CloudFront distribusi

Buat distribusi dari bucket Amazon S3. Dalam contoh berikut, parameter opsional dikomentari, tetapi nilai default ditampilkan. Untuk menambahkan penyesuaian ke distribusi Anda, batalkan komentar nilai dan parameter di dalamnya. `$distribution`

Untuk membuat CloudFront distribusi, gunakan CreateDistributionoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);

        $message = '';

        if (isset($result['Distribution']['Id'])) {
            $message = 'Distribution created with the ID of ' .
                $result['Distribution']['Id'];
```

```
        }

        $message .= ' and an effective URI of ' .
            $result['@metadata']['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'amzn-s3-demo-bucket.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
    $defaultCacheBehavior = [
        'AllowedMethods' => [
            'CachedMethods' => [
                'Items' => ['HEAD', 'GET'],
                'Quantity' => 2
            ],
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Compress' => false,
        'DefaultTTL' => 0,
        'FieldLevelEncryptionId' => '',
        'ForwardedValues' => [
            'Cookies' => [
                'Forward' => 'none'
            ],
            'Headers' => [
                'Quantity' => 0
            ],
            'QueryString' => false,
            'QueryStringCacheKeys' => [
                'Quantity' => 0
            ]
        ],
        'LambdaFunctionAssociations' => ['Quantity' => 0],
        'MaxTTL' => 0,
        'MinTTL' => 0,
```

```
        'SmoothStreaming' => false,
        'TargetOriginId' => $originName,
        'TrustedSigners' => [
            'Enabled' => false,
            'Quantity' => 0
        ],
        'ViewerProtocolPolicy' => 'allow-all'
    ];
    $enabled = false;
    $origin = [
        'Items' => [
            [
                'DomainName' => $s3BucketURL,
                'Id' => $originName,
                'OriginPath' => '',
                'CustomHeaders' => ['Quantity' => 0],
                'S3OriginConfig' => ['OriginAccessIdentity' => '']
            ]
        ],
        'Quantity' => 1
    ];
    $distribution = [
        'CallerReference' => $callerReference,
        'Comment' => $comment,
        'DefaultCacheBehavior' => $defaultCacheBehavior,
        'Enabled' => $enabled,
        'Origins' => $origin
    ];

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.
// createsTheS3Distribution();
```

Ambil distribusi CloudFront

Untuk mengambil status dan detail CloudFront distribusi tertentu, gunakan GetDistributionoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
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();
```

Daftar CloudFront distribusi

Dapatkan daftar CloudFront distribusi yang ada di AWS Wilayah yang ditentukan dari akun Anda saat ini menggunakan ListDistributionsoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
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();
```

Perbarui CloudFront distribusi

Memperbarui CloudFront distribusi mirip dengan membuat distribusi. Namun, saat Anda memperbarui distribusi, lebih banyak bidang diperlukan dan semua nilai harus disertakan. Untuk membuat perubahan pada distribusi yang ada, sebaiknya Anda mengambil distribusi yang ada terlebih dahulu, dan memperbarui nilai yang ingin Anda ubah dalam `$distribution` array.

Untuk memperbarui CloudFront distribusi tertentu, gunakan UpdateDistributionoperasi.

Impor

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

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

Kode Sampel

```
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();
```

Nonaktifkan CloudFront distribusi

Untuk menonaktifkan atau menghapus distribusi, ubah statusnya dari disebarkan menjadi dinonaktifkan.

Untuk menonaktifkan CloudFront distribusi yang ditentukan, gunakan DisableDistributionoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
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();
```

Hapus CloudFront distribusi

Setelah distribusi dalam status dinonaktifkan, Anda dapat menghapus distribusi.

Untuk menghapus CloudFront distribusi tertentu, gunakan DeleteDistributionoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
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();
```

Mengelola CloudFront pembatalan Amazon menggunakan CloudFront API dan Versi 3 AWS SDK for PHP

Amazon CloudFront menyimpan salinan file statis dan dinamis di lokasi edge di seluruh dunia. Untuk menghapus atau memperbarui file di semua lokasi tepi, buat pembatalan untuk setiap file atau untuk sekelompok file.

Setiap bulan kalender, 1.000 pembatalan pertama Anda gratis. Untuk mempelajari lebih lanjut tentang menghapus konten dari lokasi CloudFront tepi, lihat Membatalkan File.

Contoh berikut menunjukkan cara:

- Buat pembatalan distribusi menggunakan. CreateInvalidation
- Dapatkan pembatalan distribusi menggunakan. GetInvalidation
- Daftar distribusi menggunakan ListInvalidations.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan Amazon CloudFront, lihat Panduan CloudFront Pengembang Amazon.

Buat pembatalan distribusi

Buat CloudFront pembatalan distribusi dengan menentukan lokasi jalur untuk file yang perlu Anda hapus. Contoh ini membatalkan semua file dalam distribusi, tetapi Anda dapat mengidentifikasi file tertentu di bawah. `Items`

Untuk membuat pembatalan CloudFront distribusi, gunakan operasi. CreateInvalidation

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
function createInvalidation(
    $cloudFrontClient,
    $distributionId,
    $callerReference,
    $paths,
    $quantity
) {
    try {
        $result = $cloudFrontClient->createInvalidation([
            'DistributionId' => $distributionId,
            'InvalidationBatch' => [
                'CallerReference' => $callerReference,
                'Paths' => [
                    'Items' => $paths,
                    'Quantity' => $quantity,
                ],
            ]
        ]);

        $message = '';

        if (isset($result['Location'])) {
            $message = 'The invalidation location is: ' . $result['Location'];
        }
```

```
        $message .= ' and the effective URI is ' . $result['@metadata']
['effectiveUri'] . '.';

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheInvalidation()
{
    $distributionId = 'E17G7YNEXAMPLE';
    $callerReference = 'my-unique-value';
    $paths = ['/*'];
    $quantity = 1;

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createInvalidation(
        $cloudFrontClient,
        $distributionId,
        $callerReference,
        $paths,
        $quantity
    );
}

// Uncomment the following line to run this code in an AWS account.
// createTheInvalidation();
```

Dapatkan pembatalan distribusi

Untuk mengambil status dan detail tentang pembatalan CloudFront distribusi, gunakan operasi. GetInvalidation

Impor

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

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

Kode Sampel

```
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();
```

Daftar pembatalan distribusi

Untuk mencantumkan semua CloudFront pembatalan distribusi saat ini, gunakan operasi. [ListInvalidations]

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
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();
```

Menandatangani Amazon CloudFront URLs dengan AWS SDK for PHP Versi 3

Ditandatangani URLs memungkinkan Anda memberi pengguna akses ke konten pribadi Anda. URL yang ditandatangani mencakup informasi tambahan (misalnya, waktu kedaluwarsa) yang memberi Anda kontrol lebih besar atas akses ke konten Anda. Informasi tambahan ini muncul dalam pernyataan kebijakan, yang didasarkan pada kebijakan terekam atau kebijakan pabean. Untuk informasi tentang cara mengatur distribusi pribadi dan alasan Anda harus menandatangani URLs, lihat Menayangkan Konten Pribadi melalui Amazon CloudFront di Panduan CloudFront Pengembang Amazon.

- Buat CloudFront URL Amazon yang ditandatangani menggunakan getSignedurl.
- Buat CloudFront cookie Amazon yang ditandatangani menggunakan getSignedCookie.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensyal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan Amazon CloudFront, lihat Panduan CloudFront Pengembang Amazon.

Penandatanganan CloudFront URLs untuk distribusi pribadi

Anda dapat menandatangani URL menggunakan CloudFront klien di SDK. Pertama, Anda harus membuat `CloudFrontClient` objek. Anda dapat menandatangani CloudFront URL untuk sumber daya video menggunakan kebijakan kalengan atau kustom.

Impor

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

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

Kode Sampel

```
function signPrivateDistribution(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
```

```
    }
}

function signAPrivateDistribution()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo signPrivateDistribution(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistribution();
```

Menggunakan kebijakan kustom saat membuat CloudFront URLs

Untuk menggunakan kebijakan kustom, berikan `policy` kunci sebagai gantinya`expires`.

Impor

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

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

Kode Sampel

```
function signPrivateDistributionPolicy(
```

```
    $cloudFrontClient,
    $resourceKey,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedUrl([
            'url' => $resourceKey,
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function signAPrivateDistributionPolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "$resourceKey",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
                "DateLessThan": {"AWS:EpochTime": $expires}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
```

```
    ]);

    echo signPrivateDistributionPolicy(
        $cloudFrontClient,
        $resourceKey,
        $customPolicy,
        $privateKey,
        $keyPairId
    );
}

// Uncomment the following line to run this code in an AWS account.
// signAPrivateDistributionPolicy();
```

Menggunakan URL yang CloudFront ditandatangani

Bentuk URL yang ditandatangani berbeda, tergantung pada apakah URL yang Anda tandatangani menggunakan skema “HTTP” atau “RTMP”. Dalam kasus “HTTP”, URL lengkap dan absolut dikembalikan. Untuk “RTMP”, hanya URL relatif yang dikembalikan untuk kenyamanan Anda. Ini karena beberapa pemain memerlukan host dan jalur yang akan disediakan sebagai parameter terpisah.

Contoh berikut menunjukkan bagaimana Anda bisa menggunakan URL yang ditandatangani untuk membuat halaman web yang menampilkan video menggunakan. JWPlayer Jenis teknik yang sama akan berlaku untuk pemain lain seperti FlowPlayer, tetapi memerlukan kode sisi klien yang berbeda.

```
<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>
```

Menandatangani CloudFront cookie untuk distribusi pribadi

Sebagai alternatif untuk ditandatangani URLs, Anda juga dapat memberikan klien akses ke distribusi pribadi melalui cookie yang ditandatangani. Cookie yang ditandatangani memungkinkan Anda untuk menyediakan akses ke beberapa file terbatas, seperti semua file untuk video dalam format HLS atau semua file di area pelanggan situs web. Untuk informasi selengkapnya tentang alasan Anda mungkin ingin menggunakan cookie yang ditandatangani alih-alih ditandatangani URLs (atau sebaliknya), lihat Memilih antara cookie yang ditandatangani URLs dan ditandatangani di Panduan CloudFront Pengembang Amazon.

Membuat cookie yang ditandatangani mirip dengan membuat URL yang ditandatangani. Satu-satunya perbedaan adalah metode yang disebut (`getSignedCookie`bukan`getSignedUrl`).

Impor

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

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

Kode Sampel

```
function signCookie(
    $cloudFrontClient,
    $resourceKey,
    $expires,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'url' => $resourceKey,
            'expires' => $expires,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
```

```
    }
}

function signACookie()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookie(
        $cloudFrontClient,
        $resourceKey,
        $expires,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Expires = 1589926678
    CloudFront-Signature = Lv1DyC2q...2HPXaQ__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookie();
```

Gunakan kebijakan khusus saat membuat CloudFront cookie

Seperti halnya`getSignedUrl`, Anda dapat memberikan `'policy'` parameter alih-alih `expires` parameter dan `url` parameter untuk menandatangani cookie dengan kebijakan khusus. Kebijakan kustom dapat berisi wildcard di kunci sumber daya. Hal ini memungkinkan Anda untuk membuat satu cookie ditandatangani untuk beberapa file.

`getSignedCookie`mengembalikan array pasangan kunci-nilai, yang semuanya harus ditetapkan sebagai cookie untuk memberikan akses ke distribusi pribadi.

Impor

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

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

Kode Sampel

```
function signCookiePolicy(
    $cloudFrontClient,
    $customPolicy,
    $privateKey,
    $keyPairId
) {
    try {
        $result = $cloudFrontClient->getSignedCookie([
            'policy' => $customPolicy,
            'private_key' => $privateKey,
            'key_pair_id' => $keyPairId
        ]);

        return $result;
    } catch (AwsException $e) {
        return [ 'Error' => $e->getAwsErrorMessage() ];
    }
}

function signACookiePolicy()
{
    $resourceKey = 'https://d13l49jEXAMPLE.cloudfront.net/my-file.txt';
    $expires = time() + 300; // 5 minutes (5 * 60 seconds) from now.
    $customPolicy = <<<POLICY
{
    "Statement": [
        {
            "Resource": "{$resourceKey}",
            "Condition": {
                "IpAddress": {"AWS:SourceIp": "{$_SERVER['REMOTE_ADDR']}/32"},
```

```
                "DateLessThan": {"AWS:EpochTime": {$expires}}
            }
        }
    ]
}
POLICY;
    $privateKey = dirname(__DIR__) . '/cloudfront/my-private-key.pem';
    $keyPairId = 'AAPKAJIKZATYYYEXAMPLE';

    $cloudFrontClient = new CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    $result = signCookiePolicy(
        $cloudFrontClient,
        $customPolicy,
        $privateKey,
        $keyPairId
    );

    /* If successful, returns something like:
    CloudFront-Policy = eyJTdGF0...fX19XX0_
    CloudFront-Signature = RowqEQWZ...N8vetw__
    CloudFront-Key-Pair-Id = AAPKAJIKZATYYYEXAMPLE
    */
    foreach ($result as $key => $value) {
        echo $key . ' = ' . $value . "\n";
    }
}

// Uncomment the following line to run this code in an AWS account.
// signACookiePolicy();
```

Kirim CloudFront cookie ke klien Guzzle

Anda juga dapat meneruskan cookie ini ke `GuzzleHttp\Cookie\CookieJar` untuk digunakan dengan klien 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');
```

Untuk informasi selengkapnya, lihat Menggunakan Cookie yang Ditandatangani di Panduan CloudFront Pengembang Amazon.

Menandatangani permintaan CloudSearch domain Amazon khusus dengan AWS SDK for PHP Versi 3

Permintaan CloudSearch domain Amazon dapat disesuaikan di luar apa yang didukung oleh AWS SDK for PHP. Jika Anda perlu membuat permintaan khusus ke domain yang dilindungi oleh autentikasi IAM, Anda dapat menggunakan penyedia kredensi dan penandatangan SDK untuk menandatangani permintaan PSR-7 apa pun.

Misalnya, jika Anda mengikuti panduan Memulai Cloud Search dan ingin menggunakan domain yang dilindungi IAM untuk Langkah 3, Anda harus menandatangani dan menjalankan permintaan Anda sebagai berikut.

Contoh berikut menunjukkan cara:

- Tanda tangani permintaan dengan protokol AWS penandatanganan menggunakan SignatureV4.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Tanda tangani permintaan CloudSearch domain Amazon

Impor

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

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

Kode Sampel

```
function searchDomain(
    $client,
    $domainName,
    $domainId,
    $domainRegion,
    $searchString
) {
    $domainPrefix = 'search-';
    $cloudSearchDomain = 'cloudsearch.amazonaws.com';
    $cloudSearchVersion = '2013-01-01';
    $searchPrefix = 'search?';

    // Specify the search to send.
    $request = new Request(
        'GET',
        "https://$domainPrefix$domainName-$domainId.$domainRegion." .
            "$cloudSearchDomain/$cloudSearchVersion/" .
            "$searchPrefix$searchString"
    );

    // Get default AWS account access credentials.
    $credentials = call_user_func(CredentialProvider::defaultProvider())->wait();

    // Sign the search request with the credentials.
    $signer = new SignatureV4('cloudsearch', $domainRegion);
    $request = $signer->signRequest($request, $credentials);

    // Send the signed search request.
    $response = $client->send($request);

    // Report the search results, if any.
    $results = json_decode($response->getBody());

    $message = '';
```

```
    if ($results->hits->found > 0) {
        $message .= 'Search results:' . "\n";

        foreach ($results->hits->hit as $hit) {
            $message .= $hit->fields->title . "\n";
        }
    } else {
        $message .= 'No search results.';
    }

    return $message;
}

function searchADomain()
{
    $domainName = 'my-search-domain';
    $domainId = '7kbitd6nyiglhdtmssxEXAMPLE';
    $domainRegion = 'us-east-1';
    $searchString = 'q=star+wars&return=title';
    $client = new Client();

    echo searchDomain(
        $client,
        $domainName,
        $domainId,
        $domainRegion,
        $searchString
    );
}

// Uncomment the following line to run this code in an AWS account.
// searchADomain();
```

CloudWatch Contoh Amazon menggunakan AWS SDK for PHP Versi 3

Amazon CloudWatch (CloudWatch) adalah layanan web yang memantau sumber daya Amazon Web Services Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan melacak metrik, yang merupakan variabel yang dapat Anda ukur untuk sumber daya dan aplikasi Anda. CloudWatch alarm mengirim pemberitahuan atau secara otomatis membuat perubahan pada sumber daya yang Anda pantau berdasarkan aturan yang Anda tetapkan.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Topik

- Bekerja dengan CloudWatch alarm Amazon dengan AWS SDK for PHP Versi 3
- Mendapatkan metrik dari Amazon CloudWatch dengan AWS SDK for PHP Versi 3
- Menerbitkan metrik khusus di Amazon CloudWatch dengan AWS SDK for PHP Versi 3
- Mengirim acara ke CloudWatch Acara Amazon dengan AWS SDK for PHP Versi 3
- Menggunakan tindakan alarm dengan CloudWatch alarm Amazon dengan AWS SDK for PHP Versi 3

Bekerja dengan CloudWatch alarm Amazon dengan AWS SDK for PHP Versi 3

CloudWatch Alarm Amazon menonton satu metrik selama periode waktu yang Anda tentukan. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan nilai metrik yang relatif terhadap ambang batas tertentu selama beberapa periode waktu.

Contoh berikut menunjukkan cara:

- Jelaskan alarm menggunakan DescribeAlarms.
- Buat alarm menggunakan PutMetricAlarm.
- Hapus alarm menggunakan DeleteAlarms.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Mendeskripsikan alarm

Impor

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

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

Kode Sampel

```
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();
```

Membuat alarm

Impor

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

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

Kode Sampel

```
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();
```

Menghapus alarm

Impor

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

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

Kode Sampel

```
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();
```

Mendapatkan metrik dari Amazon CloudWatch dengan AWS SDK for PHP Versi 3

Metrik adalah data tentang performa sistem Anda. Anda dapat mengaktifkan pemantauan terperinci dari beberapa sumber daya, seperti EC2 instans Amazon Anda, atau metrik aplikasi Anda sendiri.

Contoh berikut menunjukkan cara:

- Daftar metrik menggunakan ListMetrics.
- Ambil alarm untuk menggunakan metrik. DescribeAlarmsForMetric
- Dapatkan statistik untuk metrik tertentu menggunakan GetMetricStatistics.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Membuat daftar metrik

Impor

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

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

Kode Sampel

```
function listMetrics($cloudWatchClient)
{
    try {
        $result = $cloudWatchClient->listMetrics();

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
```

```
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
                (isset($result['Metrics'])) and
                (count($result['Metrics']) > 0)
            ) {
                $message .= "Metrics found:\n\n";

                foreach ($result['Metrics'] as $metric) {
                    $message .= 'For metric ' . $metric['MetricName'] .
                        ' in namespace ' . $metric['Namespace'] . ":\n";

                    if (
                        (isset($metric['Dimensions'])) and
                        (count($metric['Dimensions']) > 0)
                    ) {
                        $message .= "Dimensions:\n";

                        foreach ($metric['Dimensions'] as $dimension) {
                            $message .= 'Name: ' . $dimension['Name'] .
                                ', Value: ' . $dimension['Value'] . "\n";
                        }

                        $message .= "\n";
                    } else {
                        $message .= "No dimensions.\n\n";
                    }
                }
            } else {
                $message .= 'No metrics found.';
            }
        } else {
            $message .= 'No metrics found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function listTheMetrics()
{
```

```
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo listMetrics($cloudWatchClient);
}

// Uncomment the following line to run this code in an AWS account.
// listTheMetrics();
```

Ambil alarm untuk metrik

Impor

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

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

Kode Sampel

```
function describeAlarmsForMetric(
    $cloudWatchClient,
    $metricName,
    $namespace,
    $dimensions
) {
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,
            'Namespace' => $namespace,
            'Dimensions' => $dimensions
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";
```

```
            if (
                (isset($result['MetricAlarms'])) and
                (count($result['MetricAlarms']) > 0)
            ) {
                $message .= 'Matching alarms for ' . $metricName . ":\n\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No matching alarms found for ' . $metricName . '.';
            }
        } else {
            $message .= 'No matching alarms found for ' . $metricName . '.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value' => 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'amzn-s3-demo-bucket'
        ]
    ];

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarmsForMetric(
```

```
        $cloudWatchClient,
        $metricName,
        $namespace,
        $dimensions
    );
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

Ambil statistik metrik

Impor

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

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

Kode Sampel

```
function getMetricStatistics(
    $cloudWatchClient,
    $namespace,
    $metricName,
    $dimensions,
    $startTime,
    $endTime,
    $period,
    $statistics,
    $unit
) {
    try {
        $result = $cloudWatchClient->getMetricStatistics([
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'StartTime' => $startTime,
            'EndTime' => $endTime,
            'Period' => $period,
            'Statistics' => $statistics,
            'Unit' => $unit
```

```
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'For the effective URI at ' .
                $result['@metadata']['effectiveUri'] . "\n\n";

            if (
                (isset($result['Datapoints'])) and
                (count($result['Datapoints']) > 0)
            ) {
                $message .= "Datapoints found:\n\n";

                foreach ($result['Datapoints'] as $datapoint) {
                    foreach ($datapoint as $key => $value) {
                        $message .= $key . ' = ' . $value . "\n";
                    }

                    $message .= "\n";
                }
            } else {
                $message .= 'No datapoints found.';
            }
        } else {
            $message .= 'No datapoints found.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function getTheMetricStatistics()
{
    // Average number of Amazon EC2 vCPUs every 5 minutes within
    // the past 3 hours.
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Service',
            'Value' => 'EC2'
```

```
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $startTime = strtotime('-3 hours');
    $endTime = strtotime('now');
    $period = 300; // Seconds. (5 minutes = 300 seconds.)
    $statistics = ['Average'];
    $unit = 'None';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics(
        $cloudWatchClient,
        $namespace,
        $metricName,
        $dimensions,
        $startTime,
        $endTime,
        $period,
        $statistics,
        $unit
    );

    // Another example: average number of bytes of standard storage in the
    // specified Amazon S3 bucket each day for the past 3 days.

    /*
    $namespace = 'AWS/S3';
    $metricName = 'BucketSizeBytes';
```

```
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value'=> 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'amzn-s3-demo-bucket'
        ]
    ];
    $startTime = strtotime('-3 days');
    $endTime = strtotime('now');
    $period = 86400; // Seconds. (1 day = 86400 seconds.)
    $statistics = array('Average');
    $unit = 'Bytes';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
    */
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

Menerbitkan metrik khusus di Amazon CloudWatch dengan AWS SDK for PHP Versi 3

Metrik adalah data tentang performa sistem Anda. Alarm mengawasi satu metrik selama suatu periode waktu yang Anda tentukan. Ini melakukan satu atau lebih tindakan berdasarkan nilai metrik, relatif terhadap ambang batas yang diberikan selama sejumlah periode waktu.

Contoh berikut menunjukkan cara:

- Publikasikan data metrik menggunakan PutMetricData.
- Buat alarm menggunakan PutMetricAlarm.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Publikasikan data metrik

Impor

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

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

Kode Sampel

```
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();
```

Membuat alarm

Impor

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

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

Kode Sampel

```
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();
```

Mengirim acara ke CloudWatch Acara Amazon dengan AWS SDK for PHP Versi 3

CloudWatch Acara memberikan aliran peristiwa sistem yang mendekati real-time yang menjelaskan perubahan sumber daya Amazon Web Services (AWS) ke salah satu dari berbagai target. Dengan menggunakan aturan sederhana, Anda dapat mencocokkan acara dan meruteknya ke satu atau beberapa fungsi atau aliran target.

Contoh berikut menunjukkan cara:

- Buat aturan menggunakan PutRule.
- Tambahkan target ke aturan menggunakan PutTargets.
- Kirim acara khusus ke CloudWatch Acara menggunakan PutEvents.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat aturan

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

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

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

Tambahkan target ke aturan

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

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

Kirim peristiwa kustom

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

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

Menggunakan tindakan alarm dengan CloudWatch alarm Amazon dengan AWS SDK for PHP Versi 3

Gunakan tindakan alarm untuk membuat alarm yang secara otomatis menghentikan, menghentikan, reboot, atau memulihkan instans Amazon EC2 Anda. Anda dapat menggunakan tindakan berhenti atau menghentikan saat Anda tidak lagi membutuhkan instance untuk dijalankan. Anda dapat menggunakan tindakan reboot dan memulihkan untuk secara otomatis me-reboot instance tersebut.

Contoh berikut menunjukkan cara:

- Aktifkan tindakan untuk alarm tertentu menggunakan EnableAlarmActions.
- Nonaktifkan tindakan untuk alarm tertentu menggunakan DisableAlarmActions.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Mengaktifkan tindakan alarm

Impor

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

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

Kode Sampel

```
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();
```

Menonaktifkan tindakan alarm

Impor

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

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

Kode Sampel

```
function disableAlarmActions($cloudWatchClient, $alarmNames)
{
    try {
        $result = $cloudWatchClient->disableAlarmActions([
            'AlarmNames' => $alarmNames
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            return 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] .
                ', actions for any matching alarms have been disabled.';
        } else {
            return 'Actions for some matching alarms ' .
                'might not have been disabled.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function disableTheAlarmActions()
{
    $alarmNames = array('my-alarm');

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo disableAlarmActions($cloudWatchClient, $alarmNames);
}

// Uncomment the following line to run this code in an AWS account.
// disableTheAlarmActions();
```

EC2 Contoh Amazon menggunakan AWS SDK for PHP Versi 3

Amazon Elastic Compute Cloud (Amazon EC2) adalah layanan web yang menyediakan hosting server virtual di cloud. Ini dirancang untuk membuat komputasi awan skala web lebih mudah bagi pengembang dengan menyediakan kapasitas komputasi yang dapat diubah ukurannya.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Topik

- Mengelola EC2 instans Amazon menggunakan AWS SDK for PHP Versi 3
- Menggunakan alamat IP Elastis dengan Amazon EC2 dengan AWS SDK for PHP Versi 3
- Menggunakan wilayah dan zona ketersediaan untuk Amazon EC2 dengan AWS SDK for PHP Versi 3
- Bekerja dengan pasangan EC2 kunci Amazon dengan AWS SDK for PHP Versi 3
- Bekerja dengan grup keamanan di Amazon EC2 dengan AWS SDK for PHP Versi 3

Mengelola EC2 instans Amazon menggunakan AWS SDK for PHP Versi 3

Contoh berikut menunjukkan cara:

- Jelaskan EC2 contoh Amazon menggunakan DescribeInstances.
- Aktifkan pemantauan terperinci untuk instance yang sedang berjalan menggunakan MonitorInstances.
- Nonaktifkan pemantauan untuk instance yang sedang berjalan menggunakan UnmonitorInstances.
- Mulai AMI yang didukung Amazon EBS-Backed yang sebelumnya Anda hentikan, gunakan. StartInstances
- Hentikan penggunaan instans yang didukung Amazon EBS. StopInstances
- Minta reboot dari satu atau beberapa instance menggunakan RebootInstances.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Menjelaskan instans

Impor

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

use Aws\Ec2\Ec2Client;
```

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
```

```
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
}
```

Aktifkan dan nonaktifkan pemantauan

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

$monitorInstance = 'ON';

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

var_dump($result);
```

Memulai dan menghentikan sebuah instance

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$action = 'START';

$instanceIds = ['InstanceID1', 'InstanceID2'];

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

var_dump($result);
```

Menyalakan ulang instans

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
```

```
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$instanceIds = ['InstanceID1', 'InstanceID2'];

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

var_dump($result);
```

Menggunakan alamat IP Elastis dengan Amazon EC2 dengan AWS SDK for PHP Versi 3

Alamat IP elastis adalah alamat IP statis yang dirancang untuk komputasi awan dinamis. Alamat IP Elastis dikaitkan dengan alamat IP Anda Akun AWS. Ini adalah alamat IP publik, yang dapat dijangkau dari internet. Jika instans Anda tidak memiliki alamat IP publik, Anda dapat mengaitkan alamat IP Elastis dengan instans Anda untuk mengaktifkan komunikasi dengan internet.

Contoh berikut menunjukkan cara:

- Jelaskan satu atau lebih contoh Anda menggunakan DescribeInstances.
- Dapatkan alamat IP Elastis menggunakan AllocateAddress.
- Kaitkan alamat IP Elastis dengan instance menggunakan AssociateAddress.
- Lepaskan alamat IP Elastis menggunakan ReleaseAddress.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Jelaskan sebuah contoh

Impor

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

use Aws\Ec2\Ec2Client;
```

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
$result = $ec2Client->describeInstances();
echo "Instances: \n";
foreach ($result['Reservations'] as $reservation) {
    foreach ($reservation['Instances'] as $instance) {
        echo "InstanceId: {$instance['InstanceId']} - {$instance['State']['Name']} \n";
    }
}
```

Alokasikan dan kaitkan alamat

Impor

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

Kode Sampel

```
$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);
```

Lepaskan alamat

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$associationID = 'AssociationID';

$allocationID = 'AllocationID';

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

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

var_dump($result);
```

Menggunakan wilayah dan zona ketersediaan untuk Amazon EC2 dengan AWS SDK for PHP Versi 3

Amazon EC2 di-host di beberapa lokasi di seluruh dunia. Lokasi ini terdiri dari AWS Wilayah dan Zona Ketersediaan. Setiap Wilayah adalah wilayah geografis yang terpisah, dengan beberapa lokasi terisolasi yang dikenal sebagai Availability Zone. Amazon EC2 menyediakan kemampuan untuk menempatkan instance dan data di beberapa lokasi.

Contoh berikut menunjukkan cara:

- Jelaskan Availability Zone yang tersedia untuk Anda gunakan DescribeAvailabilityZones.
- Jelaskan AWS Wilayah yang saat ini tersedia untuk Anda gunakan DescribeRegions.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Jelaskan zona ketersediaan

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

Jelaskan wilayah

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

Bekerja dengan pasangan EC2 kunci Amazon dengan AWS SDK for PHP Versi 3

Amazon EC2 menggunakan kriptografi kunci publik untuk mengenkripsi dan mendekripsi informasi login. Kriptografi kunci publik menggunakan kunci publik untuk mengenkripsi data. Kemudian penerima menggunakan kunci pribadi untuk mendekripsi data. Kunci publik dan privat dikenal sebagai pasangan kunci.

Contoh berikut menunjukkan cara:

- Buat key pair RSA 2048-bit menggunakan. CreateKeyPair
- Hapus key pair tertentu menggunakan DeleteKeyPair.
- Jelaskan satu atau lebih pasangan kunci Anda menggunakan DescribeKeyPairs.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Membuat pasangan kunci

Impor

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

Kode Sampel

```
$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);
```

Hapus key pair

Impor

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

Kode Sampel

```
$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);
```

Jelaskan pasangan kunci

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

Bekerja dengan grup keamanan di Amazon EC2 dengan AWS SDK for PHP Versi 3

Grup EC2 keamanan Amazon bertindak sebagai firewall virtual yang mengontrol lalu lintas untuk satu atau lebih contoh. Anda menambahkan aturan ke setiap grup keamanan untuk mengizinkan lalu lintas ke atau dari instans terkait. Anda dapat melakukan modifikasi terhadap aturan-aturan untuk grup keamanan kapan saja. Aturan baru diterapkan secara otomatis ke semua instance yang terkait dengan grup keamanan.

Contoh berikut menunjukkan cara:

- Jelaskan satu atau beberapa kelompok keamanan Anda menggunakan DescribeSecurityGroups.
- Tambahkan aturan ingress ke grup keamanan menggunakan AuthorizeSecurityGroupIngress.
- Buat grup keamanan menggunakan CreateSecurityGroup.
- Hapus grup keamanan menggunakan DeleteSecurityGroup.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Jelaskan kelompok keamanan

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

Tambahkan aturan masuk

Impor

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

Kode Sampel

```
$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);
```

Membuat grup keamanan

Impor

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

Kode Sampel

```
$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';
```

Menghapus grup keamanan

Impor

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

Kode Sampel

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

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

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

var_dump($result);
```

Menandatangani permintaan pencarian OpenSearch Layanan Amazon dengan AWS SDK for PHP Versi 3

Amazon OpenSearch Service adalah layanan terkelola yang memudahkan penerapan, pengoperasian, dan skala OpenSearch Layanan Amazon, pencarian sumber terbuka yang populer, dan mesin analitik. OpenSearch Layanan menawarkan akses langsung ke Amazon OpenSearch Service API. Ini berarti bahwa pengembang dapat menggunakan alat yang mereka kenal, serta opsi

keamanan yang kuat. Banyak klien Amazon OpenSearch Service mendukung penandatanganan permintaan, tetapi jika Anda menggunakan klien yang tidak, Anda dapat menandatangani permintaan PSR-7 sewenang-wenang dengan penyedia kredensi bawaan dan penandatangan. AWS SDK for PHP

Contoh berikut menunjukkan cara:

- Tanda tangani permintaan dengan protokol AWS penandatanganan menggunakan SignatureV4.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensional Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Menandatangani permintaan OpenSearch Layanan

OpenSearch Layanan menggunakan Signature Versi 4. Ini berarti Anda harus menandatangani permintaan terhadap nama penandatanganan layanan (`es`, dalam hal ini) dan AWS Wilayah domain OpenSearch Layanan Anda. Daftar lengkap Wilayah yang didukung oleh OpenSearch Layanan dapat ditemukan di halaman AWS Wilayah dan Titik Akhir di Referensi Umum Amazon Web Services. Namun, dalam contoh ini, kami menandatangani permintaan terhadap domain OpenSearch Layanan di `us-west-2` wilayah tersebut.

Anda harus memberikan kredensi, yang dapat Anda lakukan baik dengan rantai penyedia default SDK atau dengan segala bentuk kredensial yang dijelaskan dalam Kredensial untuk Versi 3. AWS SDK for PHP Anda juga memerlukan permintaan PSR-7 (diasumsikan dalam kode di bawah ini untuk diberi nama`$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);
```

AWS Identity and Access Management contoh menggunakan AWS SDK for PHP Versi 3

AWS Identity and Access Management (IAM) adalah layanan web yang memungkinkan pelanggan Amazon Web Services untuk mengelola pengguna dan izin pengguna di. AWS Layanan ini ditargetkan pada organisasi dengan banyak pengguna atau sistem di cloud yang menggunakan AWS produk. Dengan IAM, Anda dapat mengelola pengguna secara terpusat, kredensi keamanan seperti kunci akses, dan izin yang mengontrol sumber daya mana AWS yang dapat diakses pengguna.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Topik

- Mengelola kunci akses IAM dengan AWS SDK for PHP Versi 3
- Mengelola pengguna IAM dengan AWS SDK for PHP Versi 3
- Menggunakan alias akun IAM dengan Versi 3 AWS SDK for PHP
- Bekerja dengan kebijakan IAM dengan AWS SDK for PHP Versi 3
- Bekerja dengan sertifikat server IAM dengan AWS SDK for PHP Versi 3

Mengelola kunci akses IAM dengan AWS SDK for PHP Versi 3

Pengguna membutuhkan kunci akses mereka sendiri untuk melakukan panggilan terprogram. AWS Untuk memenuhi kebutuhan ini, Anda dapat membuat, memodifikasi, melihat, atau memutar kunci akses (kunci akses IDs dan kunci akses rahasia) untuk pengguna IAM. Secara default, saat Anda membuat kunci akses, statusnya adalah Aktif. Ini berarti pengguna dapat menggunakan kunci akses untuk panggilan API.

Contoh berikut menunjukkan cara:

- Buat kunci akses rahasia dan ID kunci akses yang sesuai menggunakan CreateAccessKey.
- Mengembalikan informasi tentang kunci akses IDs yang terkait dengan pengguna IAM menggunakan ListAccessKeys.

- Mengambil informasi tentang kapan kunci akses terakhir digunakan GetAccessKeyLastUsed.
- Ubah status kunci akses dari Aktif ke Tidak Aktif, atau sebaliknya, menggunakan UpdateAccessKey.
- Hapus access key pair yang terkait dengan pengguna IAM menggunakan DeleteAccessKey.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat kunci akses

Impor

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

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

Kode Sampel

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

Daftar kunci akses

Impor

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

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

Kode Sampel

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

Mendapatkan informasi tentang penggunaan terakhir kunci akses

Impor

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

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

Kode Sampel

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

Perbarui kunci akses

Impor

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

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

Kode Sampel

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

Hapus kunci akses

Impor

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

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

Kode Sampel

```
$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());
```

```
}
```

Mengelola pengguna IAM dengan AWS SDK for PHP Versi 3

Pengguna IAM adalah entitas yang Anda buat AWS untuk mewakili orang atau layanan yang menggunakannya untuk berinteraksi AWS. Seorang pengguna di AWS terdiri dari nama dan kredensi.

Contoh berikut menunjukkan cara:

- Buat pengguna IAM baru menggunakan CreateUser.
- Daftar pengguna IAM menggunakan ListUsers.
- Perbarui pengguna IAM menggunakan UpdateUser.
- Mengambil informasi tentang pengguna IAM menggunakan. GetUser
- Hapus pengguna IAM menggunakan DeleteUser.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Mmebuat pengguna IAM

Impor

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

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

Kode Sampel

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

Daftar pengguna IAM

Impor

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

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

Kode Sampel

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

Memperbarui pengguna IAM

Impor

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

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

Kode Sampel

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

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

Dapatkan informasi tentang pengguna IAM

Impor

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

use Aws\Exception\AwsException;
```

```
use Aws\Iam\IamClient;
```

Kode Sampel

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

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

Menghapus pengguna IAM

Impor

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

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

Kode Sampel

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

try {
```

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

Menggunakan alias akun IAM dengan Versi 3 AWS SDK for PHP

Jika Anda ingin URL untuk halaman login berisi nama perusahaan atau pengenal ramah lainnya, bukan Akun AWS ID Anda, Anda dapat membuat alias untuk ID Anda. Akun AWS Jika Anda membuat Akun AWS alias, URL halaman masuk Anda berubah untuk memasukkan alias.

Contoh berikut menunjukkan cara:

- Buat alias menggunakan CreateAccountAlias.
- Buat daftar alias yang terkait dengan Akun AWS penggunaan ListAccountAliases.
- Hapus alias menggunakan DeleteAccountAlias.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Membuat alias

Impor

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

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

Kode Sampel

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

Daftar alias akun

Impor

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

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

Kode Sampel

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

Hapus alias

Impor

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

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

Kode Sampel

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

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

Bekerja dengan kebijakan IAM dengan AWS SDK for PHP Versi 3

Anda memberikan izin kepada pengguna dengan membuat kebijakan. Kebijakan adalah dokumen yang mencantumkan tindakan yang dapat dilakukan pengguna dan sumber daya yang dapat memengaruhi tindakan tersebut. Secara default, tindakan atau sumber daya apa pun yang tidak

diizinkan secara eksplisit ditolak. Kebijakan dapat dibuat dan dilampirkan ke pengguna, grup pengguna, peran yang diambil oleh pengguna, dan sumber daya.

Contoh berikut menunjukkan cara:

- Buat kebijakan terkelola menggunakan CreatePolicy.
- Lampirkan kebijakan ke peran yang digunakan AttachRolePolicy.
- Lampirkan kebijakan ke pengguna yang menggunakan AttachUserPolicy.
- Lampirkan kebijakan ke grup yang menggunakan AttachGroupPolicy.
- Hapus kebijakan peran menggunakan DetachRolePolicy.
- Hapus kebijakan pengguna menggunakan DetachUserPolicy.
- Hapus kebijakan grup menggunakan DetachGroupPolicy.
- Menghapus kebijakan terkelola menggunakan DeletePolicy.
- Menghapus kebijakan peran menggunakan DeleteRolePolicy.
- Hapus kebijakan pengguna menggunakan DeleteUserPolicy.
- Hapus kebijakan grup menggunakan DeleteGroupPolicy.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat kebijakan

Impor

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

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

Kode Sampel

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

Lampirkan kebijakan ke peran

Impor

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

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

Kode Sampel

```
$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());
```

```
}
```

Lampirkan kebijakan ke pengguna

Impor

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

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

Kode Sampel

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

Lampirkan kebijakan ke grup

Impor

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

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

Kode Sampel

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

Lepaskan kebijakan pengguna

Impor

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

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

Kode Sampel

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

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

Lepaskan kebijakan grup

Impor

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

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

Kode Sampel

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

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

Menghapus kebijakan

Impor

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

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

Kode Sampel

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

Menghapus kebijakan peran

Impor

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

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

Kode Sampel

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

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

```
}
```

Menghapus kebijakan pengguna

Impor

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

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

Kode Sampel

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

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

Menghapus kebijakan grup

Impor

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

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

Kode Sampel

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

Bekerja dengan sertifikat server IAM dengan AWS SDK for PHP Versi 3

Untuk mengaktifkan koneksi HTTPS ke situs web atau aplikasi Anda AWS, Anda memerlukan sertifikat SSL/TLS server. Untuk menggunakan sertifikat yang Anda peroleh dari penyedia eksternal dengan situs web atau aplikasi Anda AWS, Anda harus mengunggah sertifikat ke IAM atau mengimpornya ke dalam AWS Certificate Manager.

Contoh berikut menunjukkan cara:

- Buat daftar sertifikat yang disimpan dalam IAM menggunakan ListServerCertificates.
- Ambil informasi tentang sertifikat yang menggunakan GetServerCertificate.
- Perbarui sertifikat menggunakan UpdateServerCertificate.
- Hapus sertifikat menggunakan DeleteServerCertificate.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Daftar sertifikat server

Impor

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

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

Kode Sampel

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

Mengambil sertifikat server

Impor

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

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

Kode Sampel

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

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

Memperbarui sertifikat server

Impor

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

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

Kode Sampel

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

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

Hapus sertifikat server

Impor

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

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

Kode Sampel

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

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

AWS Key Management Service contoh menggunakan AWS SDK for PHP Versi 3

AWS Key Management Service (AWS KMS) adalah layanan terkelola yang memudahkan Anda membuat dan mengontrol kunci enkripsi yang digunakan untuk mengenkripsi data Anda. Untuk informasi selengkapnya AWS KMS, lihat dokumentasi Amazon KMS. Apakah Anda menulis aplikasi PHP yang aman atau mengirim data ke AWS layanan lain, AWS KMS membantu Anda mempertahankan kendali atas siapa yang dapat menggunakan kunci Anda dan mendapatkan akses ke data terenkripsi Anda.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Topik

- Bekerja dengan kunci menggunakan AWS KMS API dan AWS SDK for PHP Versi 3
- Mengenkripsi dan mendekripsi kunci AWS KMS data menggunakan Versi 3 AWS SDK for PHP
- Bekerja dengan kebijakan AWS KMS utama menggunakan AWS SDK for PHP Versi 3
- Bekerja dengan hibah menggunakan AWS KMS API dan AWS SDK for PHP versi 3
- Bekerja dengan alias menggunakan AWS KMS API dan AWS SDK for PHP Versi 3

Bekerja dengan kunci menggunakan AWS KMS API dan AWS SDK for PHP Versi 3

Sumber daya utama di AWS Key Management Service (AWS KMS) adalah AWS KMS keys. Anda dapat menggunakan kunci KMS untuk mengenkripsi data Anda.

Contoh berikut menunjukkan cara:

- Buat kunci KMS pelanggan menggunakan CreateKey.
- Hasilkan kunci data menggunakan GenerateDataKey.
- Lihat tombol KMS menggunakan DescribeKey.
- Dapatkan kunci IDs dan kunci ARNS dari tombol KMS menggunakan. ListKeys
- Aktifkan tombol KMS menggunakan EnableKey.
- Nonaktifkan tombol KMS menggunakan DisableKey.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan AWS Key Management Service (AWS KMS), lihat Panduan AWS KMS Pengembang.

Buat kunci KMS

Untuk membuat kunci KMS, gunakan CreateKeyoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menghasilkan kunci data

Untuk menghasilkan kunci enkripsi data, gunakan GenerateDataKeyoperasi. Operasi ini mengembalikan plaintext dan salinan terenkripsi dari kunci data yang dibuatnya. Tentukan AWS KMS key di bawah mana untuk menghasilkan kunci data.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Lihat kunci KMS

Untuk mendapatkan informasi terperinci tentang kunci KMS, termasuk nama sumber daya Amazon (ARN) kunci KMS dan status kunci, gunakan operasi. DescribeKey

`DescribeKey`tidak mendapatkan alias. Untuk mendapatkan alias, gunakan ListAliasesoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Dapatkan ID kunci dan kunci kunci ARNs KMS

Untuk mendapatkan ID dan ARN dari kunci KMS, gunakan operasi. ListAliases

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Aktifkan kunci KMS

Untuk mengaktifkan kunci KMS yang dinonaktifkan, gunakan EnableKeyoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Nonaktifkan tombol KMS

Untuk menonaktifkan kunci KMS, gunakan DisableKeyoperasi. Menonaktifkan kunci KMS mencegahnya digunakan.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengenkripsi dan mendekripsi kunci AWS KMS data menggunakan Versi 3 AWS SDK for PHP

Kunci data adalah kunci enkripsi yang dapat Anda gunakan untuk mengenkripsi data, termasuk data dalam jumlah besar dan kunci enkripsi data lainnya.

Anda dapat menggunakan AWS Key Management Service an (AWS KMS) AWS KMS keyuntuk menghasilkan, mengenkripsi, dan mendekripsi kunci data.

Contoh berikut menunjukkan cara:

- Enkripsi kunci data menggunakan Encrypt.
- Dekripsi kunci data menggunakan Dekripsi.
- Enkripsi ulang kunci data dengan kunci KMS baru menggunakan. ReEncrypt

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan AWS Key Management Service (AWS KMS), lihat Panduan AWS KMS Pengembang.

Enkripsi

Operasi Encrypt dirancang untuk mengenkripsi kunci data, tetapi tidak sering digunakan. GenerateDataKeyWithoutPlaintextOperasi GenerateDataKeydan mengembalikan kunci data terenkripsi. Anda dapat menggunakan `Encypt` metode ini saat memindahkan data terenkripsi ke AWS Wilayah baru dan ingin mengenkripsi kunci datanya dengan menggunakan kunci KMS di Wilayah baru.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Dekripsi

Untuk mendekripsi kunci data, gunakan operasi Decrypt.

`ciphertextBlob`Yang Anda tentukan harus menjadi nilai `CiphertextBlob` bidang dari GenerateDataKey, GenerateDataKeyWithoutPlaintext, atau Enkripsi respons.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Enkripsi ulang

Untuk mendekripsi kunci data terenkripsi, dan kemudian segera mengenkripsi ulang kunci data di bawah kunci KMS yang berbeda, gunakan operasi. ReEncrypt Operasi dilakukan sepenuhnya di sisi server di dalam AWS KMS, sehingga mereka tidak pernah mengekspos plaintext Anda di luar. AWS KMS

`ciphertextBlob`Yang Anda tentukan harus menjadi nilai `CiphertextBlob` bidang dari GenerateDataKey, GenerateDataKeyWithoutPlaintext, atau Enkripsi respons.

Impor

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

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

Kode Sampel

```
$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";
}
```

Bekerja dengan kebijakan AWS KMS utama menggunakan AWS SDK for PHP Versi 3

Saat Anda membuat AWS KMS key, Anda menentukan siapa yang dapat menggunakan dan mengelola kunci KMS itu. Izin ini disertakan dalam dokumen yang disebut kebijakan kunci. Anda dapat menggunakan kebijakan kunci untuk menambah, menghapus, atau mengubah izin kapan saja untuk kunci KMS yang dikelola pelanggan, tetapi Anda tidak dapat mengedit kebijakan kunci untuk kunci KMS AWS terkelola. Untuk informasi selengkapnya, lihat Otentikasi dan kontrol akses untuk AWS KMS.

Contoh berikut menunjukkan cara:

- Buat daftar nama-nama kebijakan utama yang menggunakan ListKeyPolicies.
- Dapatkan kebijakan utama menggunakan GetKeyPolicy.

- Tetapkan kebijakan kunci menggunakan PutKeyPolicy.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan AWS Key Management Service (AWS KMS), lihat Panduan AWS KMS Pengembang.

Daftar semua kebijakan utama

Untuk mendapatkan nama kebijakan kunci untuk kunci KMS, gunakan `ListKeyPolicies` operasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengambil kebijakan utama

Untuk mendapatkan kebijakan kunci untuk kunci KMS, gunakan `GetKeyPolicy` operasi.

`GetKeyPolicy` memerlukan nama kebijakan. Kecuali Anda membuat kebijakan kunci saat membuat kunci KMS, satu-satunya nama kebijakan yang valid adalah default. Pelajari selengkapnya tentang kebijakan kunci default di Panduan AWS Key Management Service Pengembang.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menetapkan kebijakan utama

Untuk membuat atau mengubah kebijakan kunci untuk kunci KMS, gunakan `PutKeyPolicy` operasi.

`PutKeyPolicy` memerlukan nama kebijakan. Kecuali Anda membuat Kebijakan Kunci saat membuat kunci KMS, satu-satunya nama kebijakan yang valid adalah default. Pelajari selengkapnya tentang kebijakan kunci default di Panduan AWS Key Management Service Pengembang.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Bekerja dengan hibah menggunakan AWS KMS API dan AWS SDK for PHP versi 3

Hibah adalah mekanisme lain untuk memberikan izin. Ini adalah alternatif dari kebijakan utama. Anda dapat menggunakan hibah untuk memberikan akses jangka panjang yang memungkinkan AWS prinsipal menggunakan () yang dikelola pelanggan Anda AWS Key Management Service .AWS KMSAWS KMS keys Untuk informasi selengkapnya, lihat Hibah AWS KMS di Panduan AWS Key Management Service Pengembang.

Contoh berikut menunjukkan cara:

- Buat hibah untuk menggunakan CreateGrantkunci KMS.
- Lihat hibah untuk menggunakan ListGrantskunci KMS.
- Pensiun hibah untuk menggunakan kunci KMS. RetireGrant
- Mencabut hibah untuk menggunakan kunci KMS. RevokeGrant

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan AWS Key Management Service (AWS KMS), lihat Panduan AWS KMS Pengembang.

Buat hibah

Untuk membuat hibah untuk AWS KMS key, gunakan CreateGrantoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Lihat izin

Untuk mendapatkan informasi rinci tentang hibah pada AWS KMS key, gunakan ListGrantsoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Pensiun hibah

Untuk pensiun hibah untuk AWS KMS key, gunakan RetireGrantoperasi. Pensiun hibah untuk membersihkan setelah Anda selesai menggunakannya.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mencabut hibah

Untuk mencabut hibah ke AWS KMS key, gunakan operasi. RevokeGrant Anda dapat mencabut izin untuk secara eksplisit menolak operasi yang bergantung padanya.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
```

```
}
```

Bekerja dengan alias menggunakan AWS KMS API dan AWS SDK for PHP Versi 3

AWS Key Management Service (AWS KMS) menyediakan nama tampilan opsional untuk yang AWS KMS keydisebut alias.

Contoh berikut menunjukkan cara:

- Buat alias menggunakan CreateAlias.
- Lihat alias menggunakan ListAliases.
- Perbarui alias menggunakan UpdateAlias.
- Hapus alias menggunakan DeleteAlias.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensional Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan AWS Key Management Service (AWS KMS), lihat Panduan AWS KMS Pengembang.

Membuat alias

Untuk membuat alias untuk kunci KMS, gunakan operasi. CreateAlias Alias harus unik di akun dan AWS Wilayah. Jika Anda membuat alias untuk kunci KMS yang sudah memiliki alias, `CreateAlias` buat alias lain ke kunci KMS yang sama. Itu tidak menggantikan alias yang ada.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Lihat alias

Untuk daftar semua alias di pemanggil Akun AWS dan Wilayah AWS, gunakan operasi. ListAliases

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Perbarui alias

Untuk mengaitkan alias yang ada dengan kunci KMS yang berbeda, gunakan operasi. UpdateAlias

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Hapus alias

Untuk menghapus alias, gunakan DeleteAliasoperasi. Menghapus alias tidak berpengaruh pada kunci KMS yang mendasarinya.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Contoh Amazon Kinesis menggunakan Versi 3 AWS SDK for PHP

Amazon Kinesis adalah AWS layanan yang mengumpulkan, memproses, dan menganalisis data secara real time. Konfigurasikan aliran data Anda dengan Amazon Kinesis Data Streams atau gunakan Amazon Data Firehose untuk mengirim data ke Amazon S3, OpenSearch Layanan, Amazon Redshift, atau Splunk.

Untuk informasi selengkapnya tentang Kinesis, lihat dokumentasi Amazon Kinesis.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Topik

- Membuat aliran data menggunakan Kinesis Data Streams API dan Versi 3 AWS SDK for PHP
- Mengelola pecahan data menggunakan Kinesis Data Streams API dan Versi 3 AWS SDK for PHP
- Membuat aliran pengiriman menggunakan Firehose API dan Versi 3 AWS SDK for PHP

Membuat aliran data menggunakan Kinesis Data Streams API dan Versi 3 AWS SDK for PHP

Amazon Kinesis Data Streams memungkinkan Anda mengirim data waktu nyata. Buat produsen data dengan Kinesis Data Streams yang mengirimkan data ke tujuan yang dikonfigurasi setiap kali Anda menambahkan data.

Untuk informasi selengkapnya, lihat Membuat dan Mengelola Streaming di Panduan Pengembang Amazon Kinesis.

Contoh berikut menunjukkan cara:

- Buat aliran data menggunakan CreateAlias.
- Dapatkan detail tentang aliran data tunggal yang menggunakan DescribeStream.
- Daftar aliran data yang ada menggunakan ListStreams.
- Kirim data ke aliran data yang ada menggunakan PutRecord.
- Hapus aliran data menggunakan DeleteStream.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang menggunakan Panduan Pengembang Amazon Kinesis, lihat Panduan Pengembang Amazon Kinesis Data Streams.

Buat aliran data menggunakan aliran data Kinesis

Buat aliran data Kinesis di mana Anda dapat mengirim informasi untuk diproses oleh Kinesis menggunakan contoh kode berikut. Pelajari selengkapnya tentang Membuat dan Memperbarui Aliran Data di Panduan Pengembang Amazon Kinesis.

Untuk membuat aliran data Kinesis, gunakan operasi. CreateStream

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengambil aliran data

Dapatkan detail tentang aliran data yang ada menggunakan contoh kode berikut. Secara default, ini mengembalikan informasi tentang 10 pecahan pertama yang terhubung ke aliran data Kinesis yang ditentukan. Ingatlah untuk memeriksa `StreamStatus` dari respons sebelum menulis data ke aliran data Kinesis.

Untuk mengambil rincian tentang aliran data Kinesis tertentu, gunakan DescribeStreamoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Buat daftar aliran data yang ada yang terhubung ke Kinesis

Buat daftar 10 aliran data pertama dari Anda Akun AWS di AWS Wilayah yang dipilih. Gunakan yang dikembalikan `HasMoreStreams` untuk menentukan apakah ada lebih banyak aliran yang terkait dengan akun Anda.

Untuk membuat daftar aliran data Kinesis Anda, gunakan operasi. ListStreams

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengirim data ke aliran data yang ada

Setelah Anda membuat aliran data, gunakan contoh berikut untuk mengirim data. Sebelum mengirim data ke sana, gunakan `DescribeStream` untuk memeriksa `StreamStatus` apakah data aktif.

Untuk menulis catatan data tunggal ke aliran data Kinesis, gunakan operasi. PutRecord Untuk menulis hingga 500 catatan ke dalam aliran data Kinesis, gunakan operasi. PutRecords

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menghapus aliran data

Contoh ini menunjukkan cara menghapus aliran data. Menghapus aliran data juga menghapus data apa pun yang Anda kirim ke aliran data. Aliran data Kinesis aktif beralih ke status DELETING hingga penghapusan aliran selesai. Saat dalam keadaan DELETING, aliran terus memproses data.

Untuk menghapus aliran data Kinesis, gunakan operasi. DeleteStream

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengelola pecahan data menggunakan Kinesis Data Streams API dan Versi 3 AWS SDK for PHP

Amazon Kinesis Data Streams memungkinkan Anda mengirim data real-time ke titik akhir. Laju aliran data tergantung pada jumlah pecahan dalam aliran Anda.

Anda dapat menulis 1.000 catatan per detik ke satu pecahan. Setiap shard juga memiliki batas upload 1 MiB per detik. Penggunaan dihitung dan dibebankan pada basis per-shard, jadi gunakan contoh ini untuk mengelola kapasitas data dan biaya streaming Anda.

Contoh berikut menunjukkan cara:

- Buat daftar pecahan dalam aliran menggunakan ListShards.
- Tambahkan atau kurangi jumlah pecahan dalam aliran menggunakan UpdateShardCount.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon Kinesis Data Streams, lihat Panduan Pengembang Amazon Kinesis Data Streams.

Daftar pecahan aliran data

Buat daftar detail hingga 100 pecahan dalam aliran tertentu.

Untuk membuat daftar pecahan dalam aliran data Kinesis, gunakan ListShardsoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
```

```
}
```

Tambahkan lebih banyak pecahan aliran data

Jika Anda membutuhkan lebih banyak pecahan aliran data, Anda dapat meningkatkan jumlah pecahan Anda saat ini. Kami menyarankan Anda menggandakan jumlah pecahan Anda saat meningkatkan. Ini membuat salinan setiap pecahan saat ini tersedia untuk meningkatkan kapasitas Anda. Anda dapat menggandakan jumlah pecahan Anda hanya dua kali dalam satu periode 24 jam.

Ingatlah bahwa penagihan untuk penggunaan Kinesis Data Streams dihitung per shard, jadi ketika permintaan menurun, kami sarankan Anda mengurangi jumlah pecahan hingga setengahnya. Saat Anda menghapus pecahan, Anda hanya dapat menurunkan jumlah pecahan menjadi setengah dari jumlah pecahan Anda saat ini.

Untuk memperbarui jumlah pecahan aliran data Kinesis, gunakan UpdateShardCountoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Membuat aliran pengiriman menggunakan Firehose API dan Versi 3 AWS SDK for PHP

Amazon Data Firehose memungkinkan Anda mengirim data real-time ke AWS layanan lain termasuk Amazon Kinesis Data Streams, Amazon S3, Layanan Amazon (OpenSearch Layanan), dan OpenSearch Amazon Redshift, atau ke Splunk. Buat produsen data dengan aliran pengiriman untuk mengirimkan data ke tujuan yang dikonfigurasi setiap kali Anda menambahkan data.

Contoh berikut menunjukkan cara:

- Buat aliran pengiriman menggunakan CreateDeliveryStream.
- Dapatkan detail tentang aliran pengiriman tunggal menggunakan DescribeDeliveryStream.
- Buat daftar aliran pengiriman Anda menggunakan ListDeliveryStreams.
- Kirim data ke aliran pengiriman menggunakan PutRecord.
- Hapus aliran pengiriman menggunakan DeleteDeliveryStream.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon Data Firehose, lihat Panduan Pengembang Amazon Kinesis Data Firehose.

Buat aliran pengiriman menggunakan aliran data Kinesis

Untuk membuat aliran pengiriman yang menempatkan data ke dalam aliran data Kinesis yang ada, gunakan operasi. CreateDeliveryStream

Hal ini memungkinkan pengembang untuk memigrasikan layanan Kinesis yang ada ke Firehose.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Buat aliran pengiriman menggunakan bucket Amazon S3

Untuk membuat aliran pengiriman yang menempatkan data ke dalam bucket Amazon S3 yang ada, gunakan operasi. CreateDeliveryStream

Berikan parameter tujuan, seperti yang dijelaskan dalam Parameter Tujuan. Kemudian pastikan Anda memberikan akses Firehose ke bucket Amazon S3, seperti yang dijelaskan dalam Berikan Akses Firehose Data Kinesis ke Tujuan Amazon S3.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Buat aliran pengiriman menggunakan OpenSearch Layanan

Untuk membuat aliran pengiriman Firehose yang menempatkan data ke dalam kluster OpenSearch Layanan, gunakan operasi. CreateDeliveryStream

Berikan parameter tujuan, seperti yang dijelaskan dalam Parameter Tujuan. Pastikan Anda memberikan akses Firehose ke kluster OpenSearch Layanan, seperti yang dijelaskan dalam Berikan Akses Firehose Data Kinesis ke Tujuan Amazon ES.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Mengambil aliran pengiriman

Untuk mendapatkan detail tentang aliran pengiriman Firehose yang ada, gunakan operasi. DescribeDeliveryStream

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Buat daftar aliran pengiriman yang ada yang terhubung ke Kinesis Data Streams

Untuk mencantumkan semua aliran pengiriman Firehose yang ada yang mengirim data ke Kinesis Data Streams, gunakan operasi. ListDeliveryStreams

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
```

```
}
```

Buat daftar aliran pengiriman yang ada yang mengirim data ke layanan lain AWS

Untuk mencantumkan semua aliran pengiriman Firehose yang ada yang mengirim data ke Amazon S3, Layanan, OpenSearch atau Amazon Redshift, atau ke Splunk, gunakan operasi. ListDeliveryStreams

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Kirim data ke aliran pengiriman Firehose yang ada

Untuk mengirim data melalui aliran pengiriman Firehose ke tujuan yang Anda tentukan, gunakan PutRecordoperasi setelah Anda membuat aliran pengiriman Firehose.

Sebelum mengirim data ke aliran pengiriman Firehose, gunakan `DescribeDeliveryStream` untuk melihat apakah aliran pengiriman aktif.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menghapus aliran pengiriman Firehose

Untuk menghapus aliran pengiriman Firehose, gunakan operasi. DeleteDeliveryStreams Ini juga menghapus data apa pun yang telah Anda kirim ke aliran pengiriman.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$firehoseClient = new Aws\Firehose\FirehoseClient([
    'profile' => 'default',
    'version' => '2015-08-04',
    'region' => 'us-east-2'
]);

$name = "my_stream_name";

try {
    $result = $firehoseClient->deleteDeliveryStream([
        'DeliveryStreamName' => $name,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

AWS Elemental MediaConvert contoh menggunakan AWS SDK for PHP Versi 3

AWS Elemental MediaConvert adalah layanan transcoding video berbasis file dengan fitur tingkat siaran. Anda dapat menggunakannya untuk membuat aset untuk siaran dan untuk pengiriman video-on-demand (VOD) di internet. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Elemental MediaConvert.

PHP API untuk AWS Elemental MediaConvert diekspos melalui kelas *`AWS.MediaConvert`*klien. Untuk informasi selengkapnya, lihat `Class: AWS.MediaConvert`di referensi API.

Buat dan kelola pekerjaan transcoding di AWS Elemental MediaConvert

Dalam contoh ini, Anda menggunakan AWS SDK for PHP Versi 3 untuk memanggil AWS Elemental MediaConvert dan membuat pekerjaan transcoding. Sebelum memulai, Anda perlu mengunggah video input ke bucket Amazon S3 yang Anda sediakan untuk penyimpanan input. Untuk daftar codec dan kontainer video masukan yang didukung, lihat Codec dan Container Input yang Didukung di Panduan Pengguna.AWS Elemental MediaConvert

Contoh berikut menunjukkan cara:

- Buat pekerjaan transcoding di AWS Elemental MediaConvert. CreateJob.
- Batalkan pekerjaan transcoding dari AWS Elemental MediaConvert antrian. CancelJob
- Ambil JSON untuk pekerjaan transcoding yang selesai. GetJob
- Ambil array JSON hingga 20 pekerjaan yang paling baru dibuat. ListJobs

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk mengakses MediaConvert klien, buat peran IAM yang memberikan AWS Elemental MediaConvert akses ke file input Anda dan bucket Amazon S3 tempat file output Anda disimpan. Untuk detailnya, lihat Mengatur Izin IAM di AWS Elemental MediaConvert Panduan Pengguna.

Buat klien

Konfigurasikan AWS SDK for PHP dengan membuat MediaConvert klien, dengan wilayah untuk kode Anda. Dalam contoh ini, wilayah diatur ke us-west-2.

Impor

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

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

Kode Sampel

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

Mendefinisikan pekerjaan transcoding sederhana

Buat JSON yang mendefinisikan parameter pekerjaan transcode.

Parameter ini dirinci. Anda dapat menggunakan AWS Elemental MediaConvert konsol untuk menghasilkan parameter pekerjaan JSON dengan memilih pengaturan pekerjaan di konsol, lalu memilih Tampilkan pekerjaan JSON di bagian bawah bagian Job. Contoh ini menunjukkan JSON untuk pekerjaan sederhana.

Kode Sampel

```
$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"
    ]
];
```

Buat pekerjaan

Setelah membuat parameter pekerjaan JSON, panggil metode createJob dengan menjalankan`AWS.MediaConvert service object`, dan meneruskan parameter. ID pekerjaan yang dibuat dikembalikan dalam data respons.

Kode Sampel

```
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";
}
```

Ambil pekerjaan

Dengan JoBid yang dikembalikan saat Anda menelepon createjob, Anda bisa mendapatkan deskripsi rinci tentang pekerjaan terbaru dalam format JSON.

Kode Sampel

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

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

Membatalkan tugas

Dengan JoBid dikembalikan saat Anda menelepon createjob, Anda dapat membatalkan pekerjaan saat masih dalam antrian. Anda tidak dapat membatalkan pekerjaan yang sudah mulai melakukan transcoding.

Kode Sampel

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->cancelJob([
        'Id' => 'JOB_ID', // REQUIRED The Job ID of the job to be cancelled.
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Daftar pekerjaan transcoding terbaru

Buat parameter JSON, termasuk nilai untuk menentukan apakah akan mengurutkan daftar dalam urutan ASCENDING, atau DESCENDING, ARN antrian pekerjaan yang akan diperiksa, dan status pekerjaan yang akan disertakan. Ini mengembalikan hingga 20 Pekerjaan. Untuk mengambil 20 pekerjaan terbaru berikutnya, gunakan string NextToken yang dikembalikan dengan hasil.

Kode Sampel

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);

try {
    $result = $mediaConvertClient->listJobs([
        'MaxResults' => 20,
        'Order' => 'ASCENDING',
        'Queue' => 'QUEUE_ARN',
        'Status' => 'SUBMITTED',
        // 'NextToken' => '<string>', //OPTIONAL To retrieve the twenty next most
 recent jobs
    ]);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Contoh Amazon S3 menggunakan Versi 3 AWS SDK for PHP

Amazon Simple Storage Service (Amazon S3) adalah layanan web yang menyediakan penyimpanan cloud yang sangat skalabel. Amazon S3 menyediakan penyimpanan objek yang mudah digunakan, dengan antarmuka layanan web sederhana untuk menyimpan dan mengambil sejumlah data dari mana saja di web.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Topik

- Membuat dan menggunakan bucket Amazon S3 dengan Versi 3 AWS SDK for PHP
- Mengelola izin akses bucket Amazon S3 dengan Versi 3 AWS SDK for PHP
- Mengonfigurasi bucket Amazon S3 dengan Versi 3 AWS SDK for PHP
- Menggunakan unggahan multipart Amazon S3 dengan Versi 3 AWS SDK for PHP
- URL pra-ditandatangani Amazon S3 dengan Versi 3 AWS SDK for PHP
- Amazon S3 pra-ditandatangani POSTs dengan Versi 3 AWS SDK for PHP
- Menggunakan bucket Amazon S3 sebagai host web statis dengan AWS SDK for PHP Versi 3
- Bekerja dengan kebijakan bucket Amazon S3 dengan Versi 3 AWS SDK for PHP
- Menggunakan titik akses S3 ARNs AWS SDK for PHP Versi 3
- Gunakan Titik Akses Multi-Wilayah Amazon S3 dengan Versi 3 AWS SDK for PHP

Membuat dan menggunakan bucket Amazon S3 dengan Versi 3 AWS SDK for PHP

Contoh berikut menunjukkan cara:

- Kembalikan daftar bucket yang dimiliki oleh pengirim permintaan yang diautentikasi menggunakan. ListBuckets
- Buat ember baru menggunakan CreateBucket.
- Tambahkan objek ke ember menggunakan PutObject.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensyal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Impor

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

use Aws\S3\S3Client;
```

Buat daftar bucket

Buat file PHP dengan kode berikut. Pertama buat layanan klien AWS.S3 yang menentukan Wilayah dan versi. AWS Kemudian panggil `listBuckets` metode, yang mengembalikan semua bucket Amazon S3 yang dimiliki oleh pengirim permintaan sebagai larik struktur Bucket.

Kode Sampel

```
$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";
}
```

Membuat bucket

Buat file PHP dengan kode berikut. Pertama buat layanan klien AWS.S3 yang menentukan Wilayah dan versi. AWS Kemudian panggil `createBucket` metode dengan array sebagai parameter. Satu-satunya bidang wajib adalah kunci 'Bucket', dengan nilai string untuk nama bucket yang akan dibuat. Namun, Anda dapat menentukan AWS Wilayah dengan bidang CreateBucketConfiguration ''. Jika berhasil, metode ini mengembalikan 'Lokasi' bucket.

Kode Sampel

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
            'Bucket' => $bucketName,
```

```
        ]);
        return 'The bucket\'s location is: ' .
            $result['Location'] . '. ' .
            'The bucket\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheBucket()
{
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2006-03-01'
    ]);

    echo createBucket($s3Client, 'amzn-s3-demo-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();
```

Masukkan benda ke dalam ember

Untuk menambahkan file ke bucket baru Anda, buat file PHP dengan kode berikut.

Di baris perintah Anda, jalankan file ini dan teruskan nama bucket tempat Anda ingin mengunggah file Anda sebagai string, diikuti oleh jalur file lengkap ke file yang akan diunggah.

Kode Sampel

```
$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";
}
```

Mengelola izin akses bucket Amazon S3 dengan Versi 3 AWS SDK for PHP

Daftar kontrol akses (ACLs) adalah salah satu opsi kebijakan akses berbasis sumber daya yang dapat Anda gunakan untuk mengelola akses ke bucket dan objek Anda. Anda dapat menggunakan ACLs untuk memberikan read/write izin dasar ke AWS akun lain. Untuk mempelajari lebih lanjut, lihat Mengelola Akses dengan ACLs.

Contoh berikut menunjukkan cara:

- Dapatkan kebijakan kontrol akses untuk bucket yang digunakan GetBucketAcl.
- Setel izin pada ember menggunakan ACLs, menggunakan PutBucketAcl.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Mendapatkan dan menetapkan kebijakan daftar kontrol akses

Impor

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

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

Kode Sampel

```

// Create a S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Gets the access control policy for a bucket
$bucket = 'amzn-s3-demo-bucket';
try {
    $resp = $s3Client->getBucketAcl([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving bucket ACL as follows: \n";
    var_dump($resp);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Sets the permissions on a bucket using access control lists (ACL).
$params = [
    'ACL' => 'public-read',
    'AccessControlPolicy' => [
        // Information can be retrieved from `getBucketAcl` response
        'Grants' => [
            [
                'Grantee' => [
                    'DisplayName' => '<string>',
                    'EmailAddress' => '<string>',
```

```
                    'ID' => '<string>',
                    'Type' => 'CanonicalUser',
                    'URI' => '<string>',
                ],
                'Permission' => 'FULL_CONTROL',
            ],
            // ...
        ],
        'Owner' => [
            'DisplayName' => '<string>',
            'ID' => '<string>',
        ],
    ],
    'Bucket' => $bucket,
];

try {
    $resp = $s3Client->putBucketAcl($params);
    echo "Succeed in setting bucket ACL.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

Mengonfigurasi bucket Amazon S3 dengan Versi 3 AWS SDK for PHP

Cross-origin resource sharing (CORS) menentukan cara aplikasi web klien yang dimuat di dalam satu domain untuk berinteraksi dengan sumber daya di domain yang berbeda. Dengan dukungan CORS di Amazon S3, Anda dapat membuat aplikasi web sisi klien yang kaya dengan Amazon S3 dan secara selektif mengizinkan akses lintas asal ke sumber daya Amazon S3 Anda.

Untuk informasi selengkapnya tentang penggunaan konfigurasi CORS dengan bucket Amazon S3, lihat Cross-Origin Resource Sharing (CORS).

Contoh berikut menunjukkan cara:

- Dapatkan konfigurasi CORS untuk menggunakan GetBucketCorsbucket.
- Atur konfigurasi CORS untuk bucket menggunakan PutBucketCors.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensyal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Dapatkan konfigurasi CORS

Buat file PHP dengan kode berikut. Pertama buat layanan klien AWS.S3, lalu panggil `getBucketCors` metode dan tentukan bucket yang konfigurasi CORS yang Anda inginkan.

Satu-satunya parameter yang diperlukan adalah nama bucket yang dipilih. Jika bucket saat ini memiliki konfigurasi CORS, konfigurasi tersebut dikembalikan oleh Amazon S3 sebagai CORSRules objek.

Impor

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

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

Kode Sampel

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

Atur konfigurasi CORS

Buat file PHP dengan kode berikut. Pertama buat layanan klien AWS.S3. Kemudian panggil `putBucketCors` metode dan tentukan bucket yang konfigurasi CORS untuk disetel, dan CORSConfiguration sebagai objek CORSRules JSON.

Impor

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

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

Kode Sampel

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

Menggunakan unggahan multipart Amazon S3 dengan Versi 3 AWS SDK for PHP

Dengan satu `PutObject` operasi, Anda dapat mengunggah objek hingga 5 GB. Namun, dengan menggunakan metode upload multipart (misalnya,,, `CreateMultipartUpload``UploadPart`,`AbortMultipartUpload`)`CompleteMultipartUpload`, Anda dapat mengunggah objek dari ukuran 5 MB hingga 5 TB.

Contoh berikut menunjukkan cara:

- Unggah objek ke Amazon S3, menggunakan. ObjectUploader
- Buat unggahan multibagian untuk objek Amazon S3 menggunakan. MultipartUploader
- Salin objek dari satu lokasi Amazon S3 ke lokasi lain menggunakan. ObjectCopier

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Pengunggah objek

Jika Anda tidak yakin apakah `PutObject` atau `MultipartUploader` yang terbaik untuk tugas itu, gunakan`ObjectUploader`. `ObjectUploader`mengunggah file besar ke Amazon S3 menggunakan `PutObject` salah satu `MultipartUploader` atau, berdasarkan ukuran payload.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

Kode Sampel

```
// Create an S3Client.
```

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

$bucket = 'your-bucket';
$key = 'my-file.zip';

// Use a stream instead of a file path.
$source = fopen('/path/to/large/file.zip', 'rb');

$uploader = new ObjectUploader(
    $s3Client,
    $bucket,
    $key,
    $source
);

do {
    try {
        $result = $uploader->upload();
        if ($result["@metadata"]["statusCode"] == '200') {
            print('<p>File successfully uploaded to ' . $result["ObjectURL"] . '.</
p>');
        }
        print($result);
        // If the SDK chooses a multipart upload, try again if there is an exception.
        // Unlike PutObject calls, multipart upload calls are not automatically
 retried.
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
} while (!isset($result));

fclose($source);
```

Konfigurasi

Konstruktor `ObjectUploader` objek menerima argumen berikut:

`$client`

`Aws\ClientInterface`Objek yang digunakan untuk melakukan transfer. Ini harus menjadi contoh dari`Aws\S3\S3Client`.

`$bucket`

(`string`, wajib) Nama ember tempat objek diunggah.

`$key`

(`string`, wajib) Kunci yang akan digunakan untuk objek yang sedang diunggah.

`$body`

(`mixed`, wajib) Data objek untuk diunggah. Bisa berupa`StreamInterface`, sumber daya aliran PHP, atau string data untuk diunggah.

`$acl`

(`string`) Daftar kontrol akses (ACL) untuk mengatur objek yang sedang diunggah. Objek bersifat pribadi secara default.

`$options`

Array asosiatif opsi konfigurasi untuk unggahan multipart. Opsi konfigurasi berikut valid:

`add_content_md5`

(`bool`) Setel ke true untuk secara otomatis menghitung MD5 checksum untuk unggahan.

`mup_threshold`

(`int`, default:`int(16777216)`) Jumlah byte untuk ukuran file. Jika ukuran file melebihi batas ini, unggahan multibagian digunakan.

`before_complete`

(`callable`) Callback untuk memanggil sebelum operasi. `CompleteMultipartUpload` Callback harus memiliki tanda tangan fungsi yang mirip dengan:`function (Aws\Command $command) {...}`. Lihat referensi CompleteMultipartUpload API untuk parameter yang dapat Anda tambahkan ke `CommandInterface` objek.

`before_initiate`

(`callable`) Callback untuk memanggil sebelum operasi. `CreateMultipartUpload` Callback harus memiliki tanda tangan fungsi yang mirip dengan:`function (Aws\Command`

`$command) {...}`. SDK memanggil callback ini jika ukuran file melebihi nilainya. `mup_threshold` Lihat referensi CreateMultipartUpload API untuk parameter yang dapat Anda tambahkan ke `CommandInterface` objek.

`before_upload`

(`callable`) Panggilan balik untuk memanggil sebelum operasi apa pun `PutObject` atau`UploadPart`. Callback harus memiliki tanda tangan fungsi yang mirip dengan:`function (Aws\Command $command) {...}`. SDK memanggil callback ini jika ukuran file kurang dari atau sama dengan nilai. `mup_threshold` Lihat referensi PutObject API untuk parameter yang dapat Anda terapkan pada `PutObject` permintaan. Untuk parameter yang berlaku untuk `UploadPart` permintaan, lihat referensi UploadPart API. SDK mengabaikan parameter apa pun yang tidak berlaku untuk operasi yang diwakili oleh objek. `CommandInterface`

`concurrency`

(`int`, default:`int(3)`) Jumlah maksimum `UploadPart` operasi bersamaan yang diizinkan selama pengunggahan multibagian.

`part_size`

(`int`, default:`int(5242880)`) Ukuran bagian, dalam byte, untuk digunakan saat melakukan pengunggahan multibagian. Nilai harus antara 5 MB dan 5 GB, inklusif.

`state`

(`Aws\Multipart\UploadState`) Objek yang mewakili status unggahan multibagian dan yang digunakan untuk melanjutkan unggahan sebelumnya. Ketika opsi ini disediakan, `$key` argumen `$bucket` and dan `part_size` opsi diabaikan.

`params`

Array asosiatif yang menyediakan opsi konfigurasi untuk setiap subperintah. Misalnya:

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

MultipartUploader

Unggahan multipart dirancang untuk meningkatkan pengalaman mengunggah objek yang lebih besar. Mereka memungkinkan Anda untuk mengunggah objek dalam beberapa bagian secara independen, dalam urutan apa pun, dan secara paralel.

Pelanggan Amazon S3 didorong untuk menggunakan unggahan multipart untuk objek yang lebih besar dari 100 MB.

MultipartUploader objek

SDK memiliki `MultipartUploader` objek khusus yang menyederhanakan proses upload multipart.

Impor

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Kode Sampel

```
$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";
}
```

Pengunggah membuat generator data bagian, berdasarkan sumber dan konfigurasi yang disediakan, dan mencoba mengunggah semua bagian. Jika beberapa bagian unggahan gagal, pengunggah terus mengunggah bagian selanjutnya hingga seluruh data sumber dibaca. Setelah itu, pengunggah

mencoba lagi untuk mengunggah bagian yang gagal atau melempar pengecualian yang berisi informasi tentang bagian yang gagal diunggah.

Menyesuaikan unggahan multipart

Anda dapat mengatur opsi kustom pada`CreateMultipartUpload`,`UploadPart`, dan `CompleteMultipartUpload` operasi yang dijalankan oleh pengunggah multibagian melalui panggilan balik yang diteruskan ke konstruktornya.

Impor

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

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

Kode Sampel

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

// Customizing a multipart upload
$source = '/path/to/large/file.zip';
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
    'before_initiate' => function (Command $command) {
        // $command is a CreateMultipartUpload operation
        $command['CacheControl'] = 'max-age=3600';
    },
    'before_upload' => function (Command $command) {
        // $command is an UploadPart operation
        $command['RequestPayer'] = 'requester';
    },
    'before_complete' => function (Command $command) {
        // $command is a CompleteMultipartUpload operation
        $command['RequestPayer'] = 'requester';
    },
```

```
]);
```

Pengumpulan sampah manual antara unggahan bagian

Jika Anda mencapai batas memori dengan unggahan besar, ini mungkin karena referensi siklik yang dihasilkan oleh SDK belum dikumpulkan oleh pengumpul sampah PHP ketika batas memori Anda tercapai. Memanggil algoritme pengumpulan secara manual di antara operasi memungkinkan siklus dikumpulkan sebelum mencapai batas itu. Contoh berikut memanggil algoritma koleksi menggunakan callback sebelum setiap bagian diunggah. Perhatikan bahwa memanggil pengumpul sampah memang datang dengan biaya kinerja, dan penggunaan optimal akan tergantung pada kasus penggunaan dan lingkungan Anda.

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

Memulihkan dari kesalahan

Ketika terjadi kesalahan selama proses pengunggahan multipart, a `MultipartUploadException` dilemparkan. Pengecualian ini menyediakan akses ke `UploadState` objek, yang berisi informasi tentang kemajuan unggahan multibagian. `UploadState`Dapat digunakan untuk melanjutkan unggahan yang gagal diselesaikan.

Impor

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Kode Sampel

```
// 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);
}
```

Melanjutkan unggahan dari `UploadState` upaya untuk mengunggah bagian yang belum diunggah. Objek negara melacak bagian yang hilang, bahkan jika mereka tidak berurutan. Pengunggah membaca atau mencari melalui file sumber yang disediakan ke rentang byte milik bagian yang masih perlu diunggah.

`UploadState`objek dapat diserialkan, sehingga Anda juga dapat melanjutkan unggahan dalam proses yang berbeda. Anda juga bisa mendapatkan `UploadState` objek, bahkan ketika Anda tidak menangani pengecualian, dengan menelepon`$uploader->getState()`.

> **Important**
>
> Aliran yang diteruskan sebagai sumber ke a tidak `MultipartUploader` diputar ulang secara otomatis sebelum mengunggah. Jika Anda menggunakan aliran alih-alih jalur file dalam loop yang mirip dengan contoh sebelumnya, setel ulang `$source` variabel di dalam `catch` blok.

Impor

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Kode Sampel

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Using stream instead of file path
$source = fopen('/path/to/large/file.zip', 'rb');
$uploader = new MultipartUploader($s3Client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

do {
    try {
        $result = $uploader->upload();
    } catch (MultipartUploadException $e) {
        rewind($source);
        $uploader = new MultipartUploader($s3Client, $source, [
            'state' => $e->getState(),
        ]);
    }
```

```
} while (!isset($result));
fclose($source);
```

Membatalkan unggahan multibagian

Unggahan multipart dapat dibatalkan dengan mengambil yang `UploadId` terkandung dalam `UploadState` objek dan meneruskannya ke. `abortMultipartUpload`

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

Unggahan multipart asinkron

Memanggil `upload()` `MultipartUploader` adalah permintaan pemblokiran. Jika Anda bekerja dalam konteks asinkron, Anda bisa mendapatkan janji untuk unggahan multibagian.

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

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

Kode Sampel

```
// 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();
```

Konfigurasi

Konstruktor `MultipartUploader` objek menerima argumen berikut:

`$client`

`Aws\ClientInterface`Objek yang digunakan untuk melakukan transfer. Ini harus menjadi contoh dari`Aws\S3\S3Client`.

`$source`

Sumber data yang sedang diunggah. Ini bisa berupa jalur atau URL (misalnya,`/path/to/file.jpg`), pegangan sumber daya (misalnya,`fopen('/path/to/file.jpg', 'r')`), atau instance aliran PSR-7.

`$config`

Array asosiatif opsi konfigurasi untuk unggahan multipart.

Opsi konfigurasi berikut valid:

`acl`

(`string`) Daftar kontrol akses (ACL) untuk mengatur objek yang sedang diunggah. Objek bersifat pribadi secara default.

`before_complete`

(`callable`) Callback untuk memanggil sebelum operasi. `CompleteMultipartUpload` Callback harus memiliki tanda tangan fungsi seperti`function (Aws\Command $command) {...}`.

`before_initiate`

(`callable`) Callback untuk memanggil sebelum operasi. `CreateMultipartUpload` Callback harus memiliki tanda tangan fungsi seperti`function (Aws\Command $command) {...}`.

`before_upload`

(`callable`) Callback untuk memanggil sebelum operasi apa pun`UploadPart`. Callback harus memiliki tanda tangan fungsi seperti`function (Aws\Command $command) {...}`.

`bucket`

(`string`, wajib) Nama ember tempat objek diunggah.

`concurrency`

(`int`, default:`int(5)`) Jumlah maksimum `UploadPart` operasi bersamaan yang diizinkan selama pengunggahan multibagian.

`key`

(`string`, wajib) Kunci yang akan digunakan untuk objek yang sedang diunggah.

`part_size`

(`int`, default:`int(5242880)`) Ukuran bagian, dalam byte, untuk digunakan saat melakukan pengunggahan multibagian. Ini harus antara 5 MB dan 5 GB, inklusif.

`state`

(`Aws\Multipart\UploadState`) Objek yang mewakili status unggahan multibagian dan yang digunakan untuk melanjutkan unggahan sebelumnya. Ketika opsi ini disediakan, `part_size` opsi `bucketkey`,, dan diabaikan.

`add_content_md5`

(`boolean`) Setel ke true untuk secara otomatis menghitung MD5 checksum untuk unggahan.

`params`

Array asosiatif yang menyediakan opsi konfigurasi untuk setiap subperintah. Misalnya:

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

Salinan multipart

Ini AWS SDK for PHP juga mencakup `MultipartCopy` objek yang digunakan dengan cara yang mirip dengan`MultipartUploader`, tetapi dirancang untuk menyalin objek antara 5 GB dan 5 TB dalam ukuran dalam Amazon S3.

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

use Aws\Exception\MultipartUploadException;
```

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

Kode Sampel

```
// Create an S3Client
$s3Client = new S3Client([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2006-03-01'
]);

//Copy objects within S3
$copier = new MultipartCopy($s3Client, '/bucket/key?versionId=foo', [
    'bucket' => 'your-bucket',
    'key' => 'my-file.zip',
]);

try {
    $result = $copier->copy();
    echo "Copy complete: {$result['ObjectURL']}\n";
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . "\n";
}
```

URL pra-ditandatangani Amazon S3 dengan Versi 3 AWS SDK for PHP

Anda dapat mengautentikasi jenis permintaan tertentu dengan meneruskan informasi yang diperlukan sebagai parameter string kueri alih-alih menggunakan header HTTP Otorisasi. Ini berguna untuk mengaktifkan akses browser pihak ketiga langsung ke data Amazon S3 pribadi Anda, tanpa memproksi permintaan. Idenya adalah untuk membuat permintaan “pra-ditandatangani” dan menyandikannya sebagai URL yang dapat digunakan pengguna lain. Selain itu, Anda dapat membatasi permintaan yang telah ditandatangani sebelumnya dengan menentukan waktu kedaluwarsa.

Buat URL yang telah ditandatangani sebelumnya untuk permintaan HTTP GET

Contoh kode berikut menunjukkan cara membuat URL pra-ditandatangani untuk permintaan HTTP GET dengan menggunakan SDK for PHP.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

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

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

echo $presignedUrl;
```

Referensi API untuk `createPresignedRequest` metode ini memberikan detail lebih lanjut.

Orang lain dapat menggunakan `$presignedUrl` nilai untuk mengambil objek dalam satu jam berikutnya. Ketika permintaan HTTP GET dibuat—menggunakan browser, misalnya—muncul ke layanan S3 bahwa panggilan tersebut berasal dari pengguna yang membuat URL yang ditandatangani tekan.

Buat URL yang telah ditandatangani sebelumnya untuk permintaan HTTP PUT

Contoh kode berikut menunjukkan cara membuat URL pra-ditandatangani untuk permintaan HTTP PUT dengan menggunakan SDK for PHP.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

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

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

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

Orang lain sekarang dapat menggunakan URL yang telah ditandatangani sebelumnya dalam permintaan HTTP PUT untuk mengunggah file:

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

// ...

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

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

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

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

Amazon S3 pra-ditandatangani POSTs dengan Versi 3 AWS SDK for PHP

Sama seperti pra-ditandatangani URLs, pra-ditandatangani POSTs memungkinkan Anda untuk memberikan akses tulis ke pengguna tanpa memberi mereka AWS kredensyal. Formulir POST yang telah ditandatangani sebelumnya dapat dibuat dengan bantuan instance PostObjectAWSS3 V4.

Contoh berikut menunjukkan cara:

- Dapatkan data untuk formulir unggah Object POST S3 menggunakan PostObjectV4.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

> **Note**
>
> `PostObjectV4`tidak bekerja dengan kredensyal yang bersumber dari. AWS IAM Identity Center

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensyal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat PostObject V4

Untuk membuat instance`PostObjectV4`, Anda harus memberikan yang berikut:

- contoh dari `Aws\S3\S3Client`
- bucket

- array asosiatif dari kolom masukan formulir
- larik kondisi kebijakan (lihat Konstruksi Kebijakan di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon)
- string waktu kedaluwarsa untuk kebijakan (opsional, satu jam secara default).

Impor

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

Kode Sampel

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;

$client = new S3Client([
    'profile' => 'default',
    'region' => 'us-east-1',
]);
$bucket = 'amzn-s3-demo-bucket10';
$starts_with = 'user/eric/';
$client->listBuckets();

// Set defaults for form input fields.
$formInputs = ['acl' => 'public-read'];

// Construct an array of conditions for policy.
$options = [
    ['acl' => 'public-read'],
    ['bucket' => $bucket],
    ['starts-with', '$key', $starts_with],
];

// Set an expiration time (optional).
$expires = '+2 hours';

$postObject = new PostObjectV4(
```

```
    $client,
    $bucket,
    $formInputs,
    $options,
    $expires
);

// Get attributes for the HTML form, for example, action, method, enctype.
$formAttributes = $postObject->getFormAttributes();

// Get attributes for the HTML form values.
$formInputs = $postObject->getFormInputs();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>PHP</title>
</head>
<body>
<form action="<?php echo $formAttributes['action'] ?>" method="<?php echo
 $formAttributes['method'] ?>"
      enctype="<?php echo $formAttributes['enctype'] ?>">
    <label id="key">
        <input hidden type="text" name="key" value="<?php echo $starts_with ?><?php
 echo $formInputs['key'] ?>"/>
    </label>
    <h3>$formInputs:</h3>
    acl: <label id="acl">
        <input readonly type="text" name="acl" value="<?php echo $formInputs['acl'] ?
>"/>
    </label><br/>
    X-Amz-Credential: <label id="credential">
        <input readonly type="text" name="X-Amz-Credential" value="<?php echo
 $formInputs['X-Amz-Credential'] ?>"/>
    </label><br/>
    X-Amz-Algorithm: <label id="algorithm">
        <input readonly type="text" name="X-Amz-Algorithm" value="<?php echo
 $formInputs['X-Amz-Algorithm'] ?>"/>
    </label><br/>
    X-Amz-Date: <label id="date">
        <input readonly type="text" name="X-Amz-Date" value="<?php echo $formInputs['X-
Amz-Date'] ?>"/>
    </label><br/><br/><br/>
```

```
    Policy: <label id="policy">
        <input readonly type="text" name="Policy" value="<?php echo
 $formInputs['Policy'] ?>"/>
    </label><br/>
    X-Amz-Signature: <label id="signature">
        <input readonly type="text" name="X-Amz-Signature" value="<?php echo
 $formInputs['X-Amz-Signature'] ?>"/>
    </label><br/><br/>
    <h3>Choose file:</h3>
    <input type="file" name="file"/> <br/><br/>
    <h3>Upload file:</h3>
    <input type="submit" name="submit" value="Upload to Amazon S3"/>
</form>
</body>
</html>
```

Menggunakan bucket Amazon S3 sebagai host web statis dengan AWS SDK for PHP Versi 3

Anda dapat menyelenggarakan situs web statis di Amazon S3. Untuk mempelajari lebih lanjut, lihat Hosting Situs Web Statis di Amazon S3.

Contoh berikut menunjukkan cara:

- Dapatkan konfigurasi situs web untuk menggunakan bucket GetBucketWebsite.
- Atur konfigurasi situs web untuk menggunakan bucket PutBucketWebsite.
- Hapus konfigurasi situs web dari ember menggunakan DeleteBucketWebsite.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensi Anda. Lihat Kredensial untuk AWS SDK for PHP Versi 3.

Dapatkan, atur, dan hapus konfigurasi situs web untuk bucket

Impor

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

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

Kode Sampel

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

// Retrieving the Bucket Website Configuration
$bucket = 'amzn-s3-demo-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
        ],
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ]
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
```

```
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Bekerja dengan kebijakan bucket Amazon S3 dengan Versi 3 AWS SDK for PHP

Anda dapat menggunakan kebijakan bucket untuk memberikan izin ke sumber daya Amazon S3 Anda. Untuk mempelajari lebih lanjut, lihat Menggunakan Kebijakan Bucket dan Kebijakan Pengguna.

Contoh berikut menunjukkan cara:

- Kembalikan kebijakan untuk bucket tertentu yang digunakan GetBucketPolicy.
- Ganti kebijakan pada ember menggunakan PutBucketPolicy.
- Menghapus kebijakan dari bucket menggunakan DeleteBucketPolicy.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Dapatkan, hapus, dan ganti kebijakan di bucket

Impor

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

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

Kode Sampel

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

$bucket = 'amzn-s3-demo-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
    echo "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deletes the policy from the bucket
try {
    $resp = $s3Client->deleteBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

```
// Replaces a policy on the bucket
try {
    $resp = $s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => 'foo policy',
    ]);
    echo "Succeed in put a policy on bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

Menggunakan titik akses S3 ARNs AWS SDK for PHP Versi 3

S3 memperkenalkan titik akses, cara baru untuk berinteraksi dengan bucket S3. Access Points dapat memiliki kebijakan dan konfigurasi unik yang diterapkan padanya, bukan langsung ke bucket. AWS SDK for PHP Ini memungkinkan Anda menggunakan titik akses ARNs di bidang bucket untuk operasi API, bukan menentukan nama bucket secara eksplisit. Rincian lebih lanjut tentang bagaimana titik akses dan ARNs pekerjaan S3 dapat ditemukan di sini. Contoh berikut menunjukkan cara:

- Gunakan GetObjectdengan titik akses ARN untuk mengambil objek dari ember.
- Gunakan PutObjectdengan titik akses ARN untuk menambahkan objek ke ember.
- Konfigurasikan klien S3 untuk menggunakan wilayah ARN alih-alih wilayah klien.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensyal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Impor

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

use Aws\S3\S3Client;
```

Dapatkan objek

Pertama buat layanan klien AWS.S3 yang menentukan wilayah dan versi. AWS Kemudian panggil `getObject` metode dengan kunci Anda dan titik akses S3 ARN di `Bucket` lapangan, yang akan mengambil objek dari ember yang terkait dengan titik akses itu.

Kode Sampel

```
$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'
]);
```

Masukkan benda ke dalam ember

Pertama buat layanan klien AWS.S3 yang menentukan Wilayah dan versi. AWS Kemudian panggil `putObject` metode dengan kunci yang diinginkan, file tubuh atau sumber, dan titik akses S3 ARN di `Bucket` lapangan, yang akan menempatkan objek di ember yang terkait dengan titik akses itu.

Kode Sampel

```
$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'
]);
```

Konfigurasikan klien S3 untuk menggunakan wilayah ARN alih-alih wilayah klien

Saat menggunakan titik akses S3 ARN dalam operasi klien S3, secara default klien akan memastikan bahwa wilayah ARN cocok dengan wilayah klien, melempar pengecualian jika tidak. Perilaku ini dapat

diubah untuk menerima wilayah ARN di atas wilayah klien dengan menyetel opsi `use_arn_region` konfigurasi ke. `true` Secara default, opsi diatur ke`false`.

Kode Sampel

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

Klien juga akan memeriksa variabel lingkungan dan opsi file konfigurasi, dalam urutan prioritas berikut:

1. Opsi klien`use_arn_region`, seperti pada contoh di atas.
2. Variabel lingkungan `AWS_S3_USE_ARN_REGION`

```
export AWS_S3_USE_ARN_REGION=true
```

1. Variabel konfigurasi `s3_use_arn_region` dalam file konfigurasi AWS bersama (secara default di`~/.aws/config`).

```
[default]
s3_use_arn_region = true
```

Gunakan Titik Akses Multi-Wilayah Amazon S3 dengan Versi 3 AWS SDK for PHP

Amazon Simple Storage Service (S3) Titik Akses Multi-Wilayah menyediakan titik akhir global untuk merutekan lalu lintas permintaan Amazon S3 di antaranya. Wilayah AWS

Anda dapat membuat Titik Akses Multi-Region menggunakan SDK for PHP, SDK AWS lain, konsol S3, atau CLI, AWS

> **Important**
>
> Untuk menggunakan Multi-Region Access Points dengan SDK for PHP, lingkungan PHP Anda harus memiliki AWS ekstensi Common Runtime AWS (CRT) diinstal.

Saat Anda membuat Titik Akses Multi-Wilayah, Amazon S3 menghasilkan Nama Sumber Daya Amazon (ARN) yang memiliki format berikut:

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

Anda dapat menggunakan ARN yang dihasilkan sebagai pengganti nama bucket untuk `getObject()` dan `putObject()` metode.

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

use Aws\S3\S3Client;

// Assign the Multi-Region Access Point to a variable and use it place of a bucket
 name.
$mrap = 'arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap';
$key = 'my-key';

$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$s3Client->putObject([
    'Bucket' => $mrap,
    'Key' => $key,
    'Body' => 'Hello World!'
]);

$result = $s3Client->getObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

echo $result['Body'] . "\n";

// Clean up.
$result = $s3Client->deleteObject([
    'Bucket' => $mrap,
    'Key' => $key
]);

$s3Client->waitUntil('ObjectNotExists', ['Bucket' => $mrap, 'Key' => $key]);
```

```
echo "Object deleted\n";
```

Mengelola rahasia menggunakan Secrets Manager API dan AWS SDK for PHP Versi 3

AWS Secrets Manager menyimpan dan mengelola rahasia bersama seperti kata sandi, kunci API, dan kredenal basis data. Dengan layanan Secrets Manager, pengembang dapat mengganti kredensi hard-code dalam kode yang diterapkan dengan panggilan tertanam ke Secrets Manager.

Secrets Manager secara native mendukung rotasi kredenal terjadwal otomatis untuk database Amazon Relational Database Service (Amazon RDS), meningkatkan keamanan aplikasi. Secrets Manager juga dapat dengan mulus memutar rahasia untuk database lain dan layanan pihak ketiga yang digunakan AWS Lambda untuk mengimplementasikan detail khusus layanan.

Contoh berikut menunjukkan cara:

- Buat rahasia menggunakan CreateSecret.
- Ambil rahasia menggunakan GetSecretValue.
- Daftar semua rahasia yang disimpan oleh Secrets Manager menggunakan ListSecrets.
- Dapatkan detail tentang penggunaan rahasia tertentu DescribeSecret.
- Perbarui rahasia tertentu menggunakan PutSecretValue.
- Siapkan rotasi rahasia menggunakan RotateSecret.
- Tandai rahasia untuk penghapusan menggunakan. DeleteSecret

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat rahasia di Secrets Manager

Untuk membuat rahasia di Secrets Manager, gunakan CreateSecretoperasi.

Dalam contoh ini, nama pengguna dan kata sandi disimpan sebagai string JSON.

Impor

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
$description = '<<Description>>';
try {
    $result = $client->createSecret([
        'Description' => $description,
        'Name' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Mengambil rahasia dari Secrets Manager

Untuk mengambil nilai rahasia yang disimpan di Secrets Manager, gunakan GetSecretValueoperasi.

Dalam contoh berikut, `secret` adalah string yang berisi nilai yang disimpan. Jika nilai untuk `username` is <<USERNAME>> dan nilai untuk `password` is<<PASSWORD>>, output dari `secret` adalah:

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

Gunakan `json_decode($secret, true)` untuk mengakses nilai array.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-east-1',
]);

$secretName = 'MySecretName';

try {
    $result = $client->getSecretValue([
        'SecretId' => $secretName,
    ]);
} catch (AwsException $e) {
    $error = $e->getAwsErrorCode();
    if ($error == 'DecryptionFailureException') {
        // Secrets Manager can't decrypt the protected secret text using the provided
 AWS KMS key.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InternalServiceErrorException') {
        // An error occurred on the server side.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidParameterException') {
        // You provided an invalid value for a parameter.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'InvalidRequestException') {
```

```
        // You provided a parameter value that is not valid for the current state of
 the resource.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
    if ($error == 'ResourceNotFoundException') {
        // We can't find the resource that you asked for.
        // Handle the exception here, and/or rethrow as needed.
        throw $e;
    }
}
// Decrypts secret using the associated KMS CMK.
// Depending on whether the secret is a string or binary, one of these fields will be
 populated.
if (isset($result['SecretString'])) {
    $secret = $result['SecretString'];
} else {
    $secret = base64_decode($result['SecretBinary']);
}
print $secret;
$secretArray = json_decode($secret, true);
$username = $secretArray['username'];
$password = $secretArray['password'];

// Your code goes here;
```

Daftar rahasia yang disimpan di Secrets Manager

Dapatkan daftar semua rahasia yang disimpan oleh Secrets Manager menggunakan ListSecretsoperasi.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Ambil detail tentang rahasia

Rahasia yang disimpan berisi metadata tentang aturan rotasi, saat terakhir diakses atau diubah, tag yang dibuat pengguna, dan Nama Sumber Daya Amazon (ARN). Untuk mendapatkan detail rahasia tertentu yang disimpan di Secrets Manager, gunakan DescribeSecretoperasi.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Perbarui nilai rahasia

Untuk menyimpan nilai rahasia terenkripsi baru di Secrets Manager, gunakan operasi. PutSecretValue

Ini menciptakan versi baru dari rahasia. Jika versi rahasia sudah ada, tambahkan `VersionStages` parameter dengan nilai `AWSCURRENT` untuk memastikan bahwa nilai baru digunakan saat mengambil nilai.

Impor

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$client = new SecretsManagerClient([
    'profile' => 'default',
    'version' => '2017-10-17',
    'region' => 'us-west-2'
]);
$secretName = 'MySecretName';
$secret = json_encode([
    "username" => getenv("SMDEMO_USERNAME"),
    "password" => getenv("SMDEMO_PASSWORD"),
]);
try {
    $result = $client->putSecretValue([
        'SecretId' => $secretName,
        'SecretString' => $secret,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    echo $e->getMessage();
    echo "\n";
}
```

Putar nilai ke rahasia yang ada di Secrets Manager

Untuk memutar nilai rahasia yang ada yang disimpan di Secrets Manager, gunakan fungsi rotasi Lambda dan operasi. RotateSecret

Sebelum Anda mulai, buat fungsi Lambda untuk memutar rahasia Anda. Katalog Contoh AWS Kode saat ini berisi beberapa contoh kode Lambda untuk memutar kredensil database Amazon RDS.

> **Note**
>
> Untuk informasi selengkapnya tentang memutar rahasia, lihat Memutar AWS Secrets Manager Rahasia Anda di AWS Secrets Manager Panduan Pengguna.

Setelah Anda mengatur fungsi Lambda Anda, konfigurasikan rotasi rahasia baru.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Ketika rotasi dikonfigurasi, Anda dapat menerapkan rotasi menggunakan RotateSecretoperasi.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Hapus rahasia dari Secrets Manager

Untuk menghapus rahasia tertentu dari Secrets Manager, gunakan DeleteSecretoperasi. Untuk mencegah penghapusan rahasia secara tidak sengaja, DeletionDate stempel secara otomatis ditambahkan ke rahasia yang menentukan jendela waktu pemulihan di mana Anda dapat membalikkan penghapusan. Jika waktu tidak ditentukan untuk jendela pemulihan, jumlah waktu default adalah 30 hari.

Impor

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Informasi terkait

AWS SDK for PHP Contoh menggunakan operasi REST berikut dari Referensi AWS Secrets Manager API:

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

Untuk informasi selengkapnya tentang penggunaan AWS Secrets Manager, lihat Panduan AWS Secrets Manager Pengguna.

Contoh Amazon SES menggunakan AWS SDK for PHP Versi 3

Amazon Simple Email Service (Amazon SES) adalah platform email yang menyediakan cara yang mudah dan hemat uang bagi Anda untuk mengirim dan menerima email menggunakan alamat email dan domain Anda sendiri. Untuk informasi selengkapnya tentang Amazon SES, lihat Panduan Pengembang Amazon SES.

AWS menawarkan dua versi layanan Amazon SES dan, dengan demikian, SDK for PHP menawarkan dua versi klien SesClient: dan Sesv2Client. Fungsionalitas klien tumpang tindih dalam banyak kasus meskipun cara metode dipanggil atau hasilnya mungkin berbeda. Keduanya APIs juga menawarkan fitur eksklusif, sehingga Anda dapat menggunakan kedua klien untuk mengakses semua fungsionalitas.

Contoh di bagian ini semua menggunakan yang asli,`SesClient`.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Topik

- Memverifikasi identitas email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3
- Membuat template email khusus menggunakan Amazon SES API dan AWS SDK for PHP Versi 3
- Mengelola filter email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3
- Membuat dan mengelola aturan email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3
- Memantau aktivitas pengiriman Anda menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

- Mengotorisasi pengirim menggunakan Amazon SES API dan Versi 3 AWS SDK for PHP

Memverifikasi identitas email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

Saat pertama kali mulai menggunakan akun Amazon Simple Email Service (Amazon SES), semua pengirim dan penerima harus diverifikasi di Wilayah AWS yang sama dengan tempat Anda mengirim email. Untuk informasi selengkapnya tentang mengirim email, lihat Mengirim Email dengan Amazon SES.

Contoh berikut menunjukkan cara:

- Verifikasi alamat email menggunakan VerifyEmailIdentity.
- Verifikasi domain email menggunakan VerifyDomainIdentity.
- Daftar semua alamat email menggunakan ListIdentities.
- Daftar semua domain email menggunakan ListIdentities.
- Hapus alamat email menggunakan DeleteIdentity.
- Hapus domain email menggunakan DeleteIdentity.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Verifikasi alamat email

Amazon SES hanya dapat mengirim email dari alamat email atau domain yang diverifikasi. Dengan memverifikasi alamat email, Anda menunjukkan bahwa Anda adalah pemilik alamat tersebut dan ingin mengizinkan Amazon SES mengirim email dari alamat tersebut.

Saat Anda menjalankan contoh kode berikut, Amazon SES mengirimkan email ke alamat yang Anda tentukan. Ketika Anda (atau penerima email) mengklik tautan di email, alamat tersebut diverifikasi.

Untuk menambahkan alamat email ke akun Amazon SES Anda, gunakan VerifyEmailIdentityoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Verifikasi domain email

Amazon SES hanya dapat mengirim email dari alamat email atau domain yang diverifikasi. Dengan memverifikasi domain, Anda menunjukkan bahwa Anda adalah pemilik domain tersebut. Saat Anda memverifikasi domain, Anda mengizinkan Amazon SES mengirim email dari alamat mana pun di domain itu.

Saat Anda menjalankan contoh kode berikut, Amazon SES memberi Anda token verifikasi. Anda harus menambahkan token ke konfigurasi DNS domain Anda. Untuk informasi selengkapnya, lihat

Memverifikasi Domain dengan Amazon SES di Panduan Pengembang Layanan Email Sederhana Amazon.

Untuk menambahkan domain pengiriman ke akun Amazon SES Anda, gunakan VerifyDomainIdentityoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Daftar alamat email

Untuk mengambil daftar alamat email yang dikirimkan di AWS Wilayah saat ini, terlepas dari status verifikasi, gunakan ListIdentitiesoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Daftar domain email

Untuk mengambil daftar domain email yang dikirimkan di AWS Wilayah saat ini, terlepas dari status verifikasi, gunakan operasi. ListIdentities

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menghapus alamat email

Untuk menghapus alamat email terverifikasi dari daftar identitas, gunakan DeleteIdentityoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Menghapus domain email

Untuk menghapus domain email terverifikasi dari daftar identitas terverifikasi, gunakan DeleteIdentityoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Membuat template email khusus menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

Amazon Simple Email Service (Amazon SES) memungkinkan Anda mengirim email yang dipersonalisasi untuk setiap penerima dengan menggunakan templat. Template mencakup baris subjek dan teks dan bagian HTML dari badan email. Bagian subjek dan tubuh juga dapat berisi nilai unik yang dipersonalisasi untuk setiap penerima.

Untuk informasi selengkapnya, lihat Mengirim Email yang Dipersonalisasi Menggunakan Amazon SES di Panduan Pengembang Layanan Email Sederhana Amazon.

Contoh berikut menunjukkan cara:

- Buat template email menggunakan CreateTemplate.
- Daftar semua template email menggunakan ListTemplates.
- Ambil template email menggunakan GetTemplate.
- Perbarui template email menggunakan UpdateTemplate.
- Hapus template email menggunakan DeleteTemplate.
- Kirim email template menggunakan SendTemplatedEmail.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Buat template email

Untuk membuat template untuk mengirim pesan email yang dipersonalisasi, gunakan CreateTemplateoperasi. Template dapat digunakan oleh akun apa pun yang berwenang untuk mengirim pesan di AWS Wilayah tempat templat ditambahkan.

> **Note**
>
> Amazon SES tidak memvalidasi HTML Anda, jadi pastikan HtmlPartitu valid sebelum mengirim email.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Dapatkan template email

Untuk melihat konten untuk template email yang ada termasuk baris subjek, isi HTML, dan teks biasa, gunakan GetTemplateoperasi. Hanya TemplateName diperlukan.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Daftar semua template email

Untuk mengambil daftar semua template email yang terkait dengan Anda Akun AWS di AWS Wilayah saat ini, gunakan ListTemplatesoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Perbarui template email

Untuk mengubah konten untuk template email tertentu termasuk baris subjek, isi HTML, dan teks biasa, gunakan UpdateTemplateoperasi.

Impor

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

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

Kode Sampel

```
$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";
}
```

Hapus template email

Untuk menghapus template email tertentu, gunakan DeleteTemplateoperasi. Yang Anda butuhkan hanyalah TemplateName.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Kirim email dengan template

Untuk menggunakan templat untuk mengirim email ke penerima, gunakan SendTemplatedEmailoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$SesClient = new Aws\Ses\SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region' => 'us-east-2'
]);

$template_name = 'Template_Name';
$sender_email = 'email_address';
$recipient_emails = ['email_address'];

try {
    $result = $SesClient->sendTemplatedEmail([
        'Destination' => [
            'ToAddresses' => $recipient_emails,
        ],
        'ReplyToAddresses' => [$sender_email],
        'Source' => $sender_email,

        'Template' => $template_name,
        'TemplateData' => '{ }'
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Mengelola filter email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

Selain mengirim email, Anda juga dapat menerima email dengan Amazon Simple Email Service (Amazon SES). Filter alamat IP memungkinkan Anda menentukan secara opsional apakah akan menerima atau menolak email yang berasal dari alamat IP atau rentang alamat IP. Untuk informasi selengkapnya, lihat Mengelola Filter Alamat IP untuk Menerima Email Amazon SES.

Contoh berikut menunjukkan cara:

- Buat filter email menggunakan CreateReceiptFilter.
- Daftar semua filter email menggunakan ListReceiptFilters.
- Hapus filter email menggunakan DeleteReceiptFilter.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensional Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Buat filter email

Untuk mengizinkan atau memblokir email dari alamat IP tertentu, gunakan CreateReceiptFilteroperasi. Berikan alamat IP atau rentang alamat dan nama unik untuk mengidentifikasi filter ini.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Daftar semua filter email

Untuk membuat daftar filter alamat IP yang terkait dengan Anda Akun AWS di AWS Wilayah saat ini, gunakan ListReceiptFiltersoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Hapus filter email

Untuk menghapus filter yang ada untuk alamat IP tertentu gunakan DeleteReceiptFilteroperasi. Berikan nama filter unik untuk mengidentifikasi filter tanda terima yang akan dihapus.

Jika Anda perlu mengubah rentang alamat yang difilter, Anda dapat menghapus filter tanda terima dan membuat yang baru.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Membuat dan mengelola aturan email menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

Selain mengirim email, Anda juga dapat menerima email dengan Amazon Simple Email Service (Amazon SES). Aturan tanda terima memungkinkan Anda menentukan apa yang Amazon SES lakukan dengan email yang diterimanya untuk alamat email atau domain yang Anda miliki. Aturan dapat mengirim email ke AWS layanan lain termasuk namun tidak terbatas pada Amazon S3, Amazon SNS, atau. AWS Lambda

Untuk selengkapnya, lihat Mengelola kumpulan aturan tanda terima untuk Menerima Email Amazon SES dan Mengelola Aturan Tanda Terima untuk Penerimaan Email Amazon SES.

Contoh berikut menunjukkan cara:

- Buat aturan tanda terima yang ditetapkan menggunakan CreateReceiptRuleSet.
- Buat aturan tanda terima menggunakan CreateReceiptRule.
- Jelaskan aturan tanda terima yang ditetapkan menggunakan DescribeReceiptRuleSet.
- Jelaskan aturan tanda terima menggunakan DescribeReceiptRule.
- Buat daftar semua set aturan tanda terima menggunakan ListReceiptRuleSets.
- Perbarui aturan tanda terima menggunakan UpdateReceiptRule.
- Hapus aturan tanda terima menggunakan DeleteReceiptRule.
- Hapus aturan tanda terima yang ditetapkan menggunakan DeleteReceiptRuleSet.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Buat set aturan tanda terima

Set aturan tanda terima berisi kumpulan aturan tanda terima. Anda harus memiliki setidaknya satu set aturan tanda terima yang terkait dengan akun Anda sebelum Anda dapat membuat aturan tanda

terima. Untuk membuat set aturan tanda terima, berikan yang unik RuleSetName dan gunakan CreateReceiptRuleSetoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Buat aturan penerimaan

Kontrol email masuk Anda dengan menambahkan aturan tanda terima ke set aturan tanda terima yang ada. Contoh ini menunjukkan cara membuat aturan tanda terima yang mengirim pesan masuk ke bucket Amazon S3, tetapi Anda juga dapat mengirim pesan ke Amazon SNS dan. AWS Lambda Untuk membuat aturan tanda terima, berikan aturan RuleSetName dan CreateReceiptRuleoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Jelaskan set aturan tanda terima

Sekali per detik, kembalikan detail set aturan tanda terima yang ditentukan. Untuk menggunakan DescribeReceiptRuleSetoperasi, berikan RuleSetName.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Jelaskan aturan tanda terima

Kembalikan detail aturan tanda terima yang ditentukan. Untuk menggunakan DescribeReceiptRuleoperasi, berikan RuleName dan RuleSetName.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Cantumkan semua set aturan tanda terima

Untuk mencantumkan kumpulan aturan tanda terima yang ada Akun AWS di bawah AWS Wilayah Anda saat ini, gunakan ListReceiptRuleSetsoperasi.

Impor

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

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

Kode Sampel

```
$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";
}
```

Perbarui aturan tanda terima

Contoh ini menunjukkan cara memperbarui aturan tanda terima yang mengirim pesan masuk ke suatu AWS Lambda fungsi, tetapi Anda juga dapat mengirim pesan ke Amazon SNS dan Amazon S3. Untuk menggunakan UpdateReceiptRuleoperasi, berikan aturan tanda terima baru dan RuleSetName.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Hapus set aturan tanda terima

Hapus kumpulan aturan tanda terima tertentu yang saat ini tidak dinonaktifkan. Ini juga menghapus semua aturan tanda terima yang dikandungnya. Untuk menghapus set aturan tanda terima, berikan RuleSetName ke DeleteReceiptRuleSetoperasi.

Impor

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

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

Kode Sampel

```
$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";
}
```

Hapus aturan tanda terima

Untuk menghapus aturan tanda terima yang ditentukan, berikan RuleName dan RuleSetName ke DeleteReceiptRuleoperasi.

Impor

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

use Aws\Exception\AwsException;
```

Kode Sampel

```
$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";
}
```

Memantau aktivitas pengiriman Anda menggunakan Amazon SES API dan AWS SDK for PHP Versi 3

Amazon Simple Email Service (Amazon SES) menyediakan metode untuk memantau aktivitas pengiriman Anda. Kami menyarankan Anda menerapkan metode ini sehingga Anda dapat melacak langkah-langkah penting, seperti bouncing akun, keluhan, dan tingkat penolakan. Tingkat pentalan dan keluhan yang terlalu tinggi dapat membahayakan kemampuan Anda untuk mengirim email menggunakan Amazon SES.

Contoh berikut menunjukkan cara:

- Periksa kuota pengiriman Anda menggunakan GetSendQuota.
- Pantau aktivitas pengiriman Anda menggunakan GetSendStatistics.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensional Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Periksa kuota pengiriman Anda

Anda dibatasi untuk mengirim hanya sejumlah pesan tertentu dalam satu periode 24 jam. Untuk memeriksa berapa banyak pesan yang masih diizinkan untuk Anda kirim, gunakan GetSendQuotaoperasi. Untuk informasi selengkapnya, lihat Mengelola Batas Pengiriman Amazon SES Anda.

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Pantau aktivitas pengiriman Anda

Untuk mengambil metrik pesan yang telah Anda kirim dalam dua minggu terakhir, gunakan operasi. GetSendStatistics Contoh ini mengembalikan jumlah upaya pengiriman, pantulan, keluhan, dan pesan yang ditolak dalam kenaikan 15 menit.

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Mengotorisasi pengirim menggunakan Amazon SES API dan Versi 3 AWS SDK for PHP

Untuk mengaktifkan AWS Identity and Access Management pengguna Akun AWS, pengguna, atau AWS layanan lain mengirim email melalui Amazon Simple Email Service (Amazon SES) atas nama Anda, Anda membuat kebijakan otorisasi pengiriman. Ini adalah dokumen JSON yang Anda lampirkan ke identitas yang Anda miliki.

Kebijakan ini secara tegas mencantumkan siapa yang Anda izinkan untuk dikirim untuk identitas itu, dan dalam kondisi apa. Semua pengirim, selain Anda dan entitas yang Anda berikan izin secara eksplisit dalam kebijakan, tidak diizinkan mengirim email. Identitas dapat tidak memiliki kebijakan, satu kebijakan, atau beberapa kebijakan yang terlampir padanya. Anda juga dapat memiliki satu kebijakan dengan beberapa pernyataan untuk mencapai efek dari beberapa kebijakan.

Untuk informasi selengkapnya, lihat Menggunakan Otorisasi Mengirim dengan Amazon SES.

Contoh berikut menunjukkan cara:

- Buat pengirim resmi menggunakan PutIdentityPolicy.
- Ambil kebijakan untuk pengirim yang berwenang menggunakan. GetIdentityPolicies
- Daftar pengirim resmi menggunakan ListIdentityPolicies.
- Mencabut izin untuk pengirim yang berwenang menggunakan. DeleteIdentityPolicy

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Untuk informasi selengkapnya tentang penggunaan Amazon SES, lihat Panduan Pengembang Amazon SES.

Buat pengirim resmi

Untuk mengizinkan orang lain Akun AWS mengirim email atas nama Anda, gunakan kebijakan identitas untuk menambah atau memperbarui otorisasi untuk mengirim email dari alamat email atau domain terverifikasi Anda. Untuk membuat kebijakan identitas, gunakan PutIdentityPolicyoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Mengambil kebijakan untuk pengirim resmi

Kembalikan kebijakan otorisasi pengiriman yang terkait dengan identitas email atau identitas domain tertentu. Untuk mendapatkan otorisasi pengiriman untuk alamat email atau domain tertentu, gunakan GetIdentityPolicyoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Daftar pengirim resmi

Untuk mencantumkan kebijakan otorisasi pengiriman yang terkait dengan identitas email atau identitas domain tertentu di AWS Wilayah saat ini, gunakan ListIdentityPoliciesoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Mencabut izin untuk pengirim yang berwenang

Hapus otorisasi pengiriman Akun AWS untuk orang lain untuk mengirim email dengan identitas email atau identitas domain dengan menghapus kebijakan identitas terkait dengan operasi. DeleteIdentityPolicy

Impor

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Kode Sampel

```
$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";
}
```

Contoh Amazon SNS menggunakan Versi 3 AWS SDK for PHP

Amazon Simple Notification Service (Amazon SNS) adalah layanan web yang mengoordinasikan dan mengelola pengiriman atau pengiriman pesan untuk berlangganan titik akhir atau klien.

Di Amazon SNS, ada dua jenis klien: penerbit (juga disebut sebagai produsen) dan pelanggan (juga disebut sebagai konsumen). Penerbit berkomunikasi secara asinkron dengan pelanggan dengan memproduksi dan mengirim pesan ke suatu topik, yang merupakan jalur akses logis dan saluran

komunikasi. Pelanggan (server web, alamat email, antrian Amazon SQS, AWS Lambda fungsi) mengkonsumsi atau menerima pesan atau pemberitahuan melalui salah satu protokol yang didukung (Amazon SQS, URLs HTTP/HTTPS, email, AWS SMS, Lambda) ketika mereka berlangganan topik.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Topik

- Mengelola topik di Amazon SNS dengan Versi 3 AWS SDK for PHP
- Mengelola langganan di Amazon SNS AWS SDK for PHP dengan Versi 3
- Mengirim pesan SMS di Amazon SNS dengan Versi 3 AWS SDK for PHP

Mengelola topik di Amazon SNS dengan Versi 3 AWS SDK for PHP

Untuk mengirim notifikasi ke Amazon Simple Queue Service (Amazon SQS) HTTP/HTTPS URLs,, email, AWS Lambda atau AWS SMS, Anda harus terlebih dahulu membuat topik yang mengelola pengiriman pesan ke pelanggan mana pun dari topik tersebut.

Dalam hal pola desain pengamat, topik seperti subjek. Setelah topik dibuat, Anda menambahkan pelanggan yang diberi tahu secara otomatis saat pesan dipublikasikan ke topik tersebut.

Pelajari lebih lanjut tentang berlangganan topik di Mengelola Langganan di Amazon AWS SDK for PHP SNS dengan Versi 3.

Contoh berikut menunjukkan cara:

- Buat topik untuk mempublikasikan notifikasi untuk digunakan CreateTopic.
- Kembalikan daftar topik pemohon menggunakan ListTopics.
- Hapus topik dan semua langganannya menggunakan DeleteTopic.
- Kembalikan semua properti topik menggunakan GetTopicAttributes.
- Izinkan pemilik topik untuk menyetel atribut topik ke nilai baru menggunakan SetTopicAttributes.

Untuk informasi selengkapnya tentang menggunakan Amazon SNS, lihat Atribut Topik Amazon SNS untuk Status Pengiriman Pesan.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredensil Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Buat topik

Untuk membuat topik, gunakan CreateTopicoperasi.

Setiap nama topik dalam Anda Akun AWS harus unik.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Buat daftar topik Anda

Untuk membuat daftar hingga 100 topik yang ada di AWS Wilayah saat ini, gunakan ListTopicsoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Hapus topik

Untuk menghapus topik yang ada dan semua langganannya, gunakan DeleteTopicoperasi.

Setiap pesan yang belum dikirim ke pelanggan juga akan dihapus.

Impor

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

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Dapatkan atribut topik

Untuk mengambil properti dari satu topik yang ada, gunakan GetTopicAttributesoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Tetapkan atribut topik

Untuk memperbarui properti dari satu topik yang ada, gunakan SetTopicAttributesoperasi.

Anda hanya dapat mengaturPolicy,DisplayName, dan DeliveryPolicy atribut.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Mengelola langganan di Amazon SNS AWS SDK for PHP dengan Versi 3

Gunakan topik Amazon Simple Notification Service (Amazon SNS) untuk mengirim notifikasi ke Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, alamat email, (), atau. AWS Server Migration Service AWS SMS AWS Lambda

Langganan dilampirkan ke topik yang mengelola pengiriman pesan ke pelanggan. Pelajari lebih lanjut cara membuat topik di Mengelola Topik di Amazon SNS dengan AWS SDK for PHP Versi 3.

Contoh berikut menunjukkan cara:

- Berlangganan topik yang ada menggunakan Berlangganan.
- Verifikasi langganan menggunakan ConfirmSubscription.
- Buat daftar langganan yang ada menggunakan ListSubscriptionsByTopic.
- Hapus langganan menggunakan Berhenti Berlangganan.
- Kirim pesan ke semua pelanggan topik menggunakan Publikasikan.

Untuk informasi selengkapnya tentang menggunakan Amazon SNS, lihat Menggunakan Amazon SNS for Messaging. System-to-System

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Berlangganan alamat email ke suatu topik

Untuk memulai langganan ke alamat email, gunakan operasi Berlangganan.

Anda dapat menggunakan metode berlangganan untuk berlangganan beberapa titik akhir yang berbeda ke topik Amazon SNS, tergantung pada nilai yang digunakan untuk parameter yang diteruskan. Ini ditunjukkan dalam contoh lain dalam topik ini.

Dalam contoh ini, titik akhir adalah alamat email. Token konfirmasi dikirim ke email ini. Verifikasi langganan dengan token konfirmasi ini dalam waktu tiga hari sejak diterimanya.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Berlangganan titik akhir aplikasi ke suatu topik

Untuk memulai langganan ke aplikasi web, gunakan operasi Berlangganan.

Anda dapat menggunakan metode berlangganan untuk berlangganan beberapa titik akhir yang berbeda ke topik Amazon SNS, tergantung pada nilai yang digunakan untuk parameter yang diteruskan. Ini ditunjukkan dalam contoh lain dalam topik ini.

Dalam contoh ini, titik akhir adalah URL. Token konfirmasi dikirim ke alamat web ini. Verifikasi langganan dengan token konfirmasi ini dalam waktu tiga hari sejak diterimanya.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Berlangganan fungsi Lambda ke suatu topik

Untuk memulai langganan ke fungsi Lambda, gunakan operasi Berlangganan.

Anda dapat menggunakan metode berlangganan untuk berlangganan beberapa titik akhir yang berbeda ke topik Amazon SNS, tergantung pada nilai yang digunakan untuk parameter yang diteruskan. Ini ditunjukkan dalam contoh lain dalam topik ini.

Dalam contoh ini, titik akhir adalah fungsi Lambda. Token konfirmasi dikirim ke fungsi Lambda ini. Verifikasi langganan dengan token konfirmasi ini dalam waktu tiga hari sejak diterimanya.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Berlangganan SMS teks ke suatu topik

Untuk mengirim pesan SMS ke beberapa nomor telepon secara bersamaan, berlangganan setiap nomor ke topik.

Untuk memulai langganan ke nomor telepon, gunakan operasi Berlangganan.

Anda dapat menggunakan metode berlangganan untuk berlangganan beberapa titik akhir yang berbeda ke topik Amazon SNS, tergantung pada nilai yang digunakan untuk parameter yang diteruskan. Ini ditunjukkan dalam contoh lain dalam topik ini.

Dalam contoh ini, titik akhir adalah nomor telepon dalam format E.164, standar untuk telekomunikasi internasional.

Token konfirmasi dikirim ke nomor telepon ini. Verifikasi langganan dengan token konfirmasi ini dalam waktu tiga hari sejak diterimanya.

Untuk cara alternatif untuk mengirim pesan SMS dengan Amazon SNS, lihat Mengirim Pesan SMS di Amazon SNS dengan AWS SDK for PHP Versi 3.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Konfirmasikan langganan ke suatu topik

Untuk benar-benar membuat langganan, pemilik titik akhir harus mengakui maksud untuk menerima pesan dari topik menggunakan token yang dikirim saat langganan dibuat pada awalnya, seperti yang dijelaskan sebelumnya. Token konfirmasi berlaku selama tiga hari. Setelah tiga hari, Anda dapat mengirim ulang token dengan membuat langganan baru.

Untuk mengonfirmasi langganan, gunakan ConfirmSubscriptionoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Daftar langganan ke suatu topik

Untuk membuat daftar hingga 100 langganan yang ada di AWS Wilayah tertentu, gunakan ListSubscriptionsoperasi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Berhenti berlangganan dari suatu topik

Untuk menghapus titik akhir yang berlangganan topik, gunakan operasi Berhenti berlangganan.

Jika langganan memerlukan otentikasi untuk dihapus, hanya pemilik langganan atau pemilik topik yang dapat berhenti berlangganan, dan tanda tangan diperlukan. AWS Jika panggilan berhenti berlangganan tidak memerlukan otentikasi dan pemohon bukan pemilik langganan, pesan pembatalan akhir akan dikirimkan ke titik akhir.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Publikasikan pesan ke topik Amazon SNS

Untuk mengirimkan pesan ke setiap titik akhir yang berlangganan topik Amazon SNS, gunakan operasi Publikasikan.

Buat objek yang berisi parameter untuk menerbitkan pesan, termasuk teks pesan dan Nama Sumber Daya Amazon (ARN) dari topik Amazon SNS.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Mengirim pesan SMS di Amazon SNS dengan Versi 3 AWS SDK for PHP

Anda dapat menggunakan Amazon Simple Notification Service (Amazon SNS) untuk mengirim pesan teks, atau pesan SMS, ke perangkat berkemampuan SMS. Anda dapat mengirim pesan

langsung ke sebuah nomor telepon, atau Anda dapat mengirim pesan ke beberapa nomor telepon sekaligus dengan berlangganan topik untuk nomor telepon tersebut dan mengirim pesan Anda ke topik tersebut.

Gunakan Amazon SNS untuk menentukan preferensi untuk pesan SMS, seperti bagaimana pengiriman Anda dioptimalkan (untuk biaya atau untuk pengiriman yang andal), batas pengeluaran bulanan Anda, cara pengiriman pesan dicatat, dan apakah akan berlangganan laporan penggunaan SMS harian. Preferensi ini diambil dan ditetapkan sebagai atribut SMS untuk Amazon SNS.

Saat Anda mengirim pesan SMS, tentukan nomor telepon menggunakan format E.164. E.164 adalah standar untuk struktur nomor telepon yang digunakan untuk telekomunikasi internasional. Nomor telepon yang mengikuti format ini dapat terdiri dari maksimum 15 digit bersama dengan prefiks tanda tambah (+) dan kode negara. Misalnya, nomor telepon AS dalam format E.164 akan muncul sebagai XXX555 +1001 0100.

Contoh berikut menunjukkan cara:

- Ambil pengaturan default untuk mengirim pesan SMS dari akun Anda menggunakan Get SMSAttributes.
- Perbarui pengaturan default untuk mengirim pesan SMS dari akun Anda menggunakan Set SMSAttributes.
- Temukan apakah pemilik nomor telepon tertentu telah memilih untuk tidak menerima pesan SMS dari akun Anda menggunakan CheckIfPhoneNumberISOptedKeluar.
- Buat daftar nomor telepon di mana pemilik telah memilih untuk tidak menerima pesan SMS dari akun Anda menggunakan ListPhoneNumberOptedOut.
- Kirim pesan teks (pesan SMS) langsung ke nomor telepon menggunakan Publikasikan.

Untuk informasi selengkapnya tentang menggunakan Amazon SNS, lihat Menggunakan Amazon SNS untuk Pemberitahuan Pengguna dengan Nomor Ponsel sebagai Pelanggan (Kirim SMS).

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Dapatkan atribut SMS

Untuk mengambil pengaturan default untuk pesan SMS, gunakan SMSAttributes operasi Dapatkan.

Contoh ini mendapatkan `DefaultSMSType` atribut. Atribut ini mengontrol apakah pesan SMS dikirim sebagai`Promotional`, yang mengoptimalkan pengiriman pesan untuk dikenakan biaya terendah, atau as`Transactional`, yang mengoptimalkan pengiriman pesan untuk mencapai keandalan tertinggi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Atur atribut SMS

Untuk memperbarui pengaturan default untuk pesan SMS, gunakan SMSAttributes operasi Set.

Contoh ini menetapkan `DefaultSMSType` atribut ke`Transactional`, yang mengoptimalkan pengiriman pesan untuk mencapai keandalan tertinggi.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Periksa apakah nomor telepon telah memilih keluar

Untuk menentukan apakah pemilik nomor telepon tertentu telah memilih untuk tidak menerima pesan SMS dari akun Anda, gunakan CheckIfPhoneNumberIsOptedOutoperasi.

Dalam contoh ini, nomor telepon dalam format E.164, standar untuk telekomunikasi internasional.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Cantumkan nomor telepon yang dipilih

Untuk mengambil daftar nomor telepon di mana pemilik telah memilih untuk tidak menerima pesan SMS dari akun Anda, gunakan operasi. ListPhoneNumbersOptedOut

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

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

Publikasikan ke pesan teks (pesan SMS)

Untuk mengirimkan pesan teks (pesan SMS) langsung ke nomor telepon, gunakan operasi Publikasikan.

Dalam contoh ini, nomor telepon dalam format E.164, standar untuk telekomunikasi internasional.

Pesan SMS dapat berisi hingga 140 byte. Batas ukuran untuk satu tindakan publikasi SMS adalah 1.600 byte.

Untuk detail selengkapnya tentang mengirim pesan SMS, lihat Mengirim Pesan SMS.

Impor

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Kode Sampel

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
```

```
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Contoh Amazon SQS menggunakan Versi 3 AWS SDK for PHP

Amazon Simple Queue Service (SQS) adalah layanan antrian pesan yang cepat, andal, terukur, dan dikelola sepenuhnya. Amazon SQS memungkinkan Anda memisahkan komponen aplikasi cloud. Amazon SQS mencakup antrian standar dengan throughput dan at-least-once pemrosesan tinggi, dan antrian FIFO yang menyediakan pengiriman FIFO (first>-in, first>-out) dan pemrosesan tepat sekali.

Semua kode contoh untuk AWS SDK for PHP Versi 3 tersedia di sini GitHub.

Topik

- Mengaktifkan polling panjang di Amazon AWS SDK for PHP SQS dengan Versi 3
- Mengelola batas waktu visibilitas di Amazon AWS SDK for PHP SQS dengan Versi 3
- Mengirim dan menerima pesan di Amazon SQS dengan AWS SDK for PHP Versi 3
- Menggunakan antrian huruf mati di Amazon SQS dengan Versi 3 AWS SDK for PHP
- Menggunakan antrian di Amazon AWS SDK for PHP SQS dengan Versi 3

Mengaktifkan polling panjang di Amazon AWS SDK for PHP SQS dengan Versi 3

Polling panjang mengurangi jumlah respons kosong dengan mengizinkan Amazon SQS menunggu waktu tertentu agar pesan tersedia dalam antrian sebelum mengirim respons. Selain itu, polling panjang menghilangkan respons kosong palsu dengan menanyakan semua server alih-alih pengambilan sampel server. Untuk mengaktifkan polling panjang, tentukan waktu tunggu bukan nol untuk pesan yang diterima. Untuk mempelajari lebih lanjut, lihat SQS Long Polling.

Contoh berikut menunjukkan cara:

- Tetapkan atribut pada antrean Amazon SQS untuk mengaktifkan polling panjang, menggunakan. SetQueueAttributes

- Ambil satu atau beberapa pesan dengan polling panjang menggunakan. ReceiveMessage
- Buat antrian polling panjang menggunakan. CreateQueue

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Tetapkan atribut pada antrian untuk mengaktifkan polling panjang

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->setQueueAttributes([
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
        'QueueUrl' => $queueUrl, // REQUIRED
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

Ambil pesan dengan polling panjang

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 20,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Buat antrian dengan polling panjang

Impor

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

```
use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```
$queueName = "QUEUE_NAME";


$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'ReceiveMessageWaitTimeSeconds' => 20
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Mengelola batas waktu visibilitas di Amazon AWS SDK for PHP SQS dengan Versi 3

Batas waktu visibilitas adalah periode waktu di mana Amazon SQS mencegah komponen konsumsi lainnya menerima dan memproses pesan. Untuk mempelajari lebih lanjut, lihat Batas Waktu Visibilitas.

Contoh berikut menunjukkan cara:

- Ubah batas waktu visibilitas pesan tertentu dalam antrian ke nilai baru, menggunakan. ChangeMessageVisibilityBatch

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Mengubah batas waktu visibilitas beberapa pesan

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage(array(
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 10,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
    ));
    $messages = $result->get('Messages');
    if ($messages != null) {
        $entries = array();
        for ($i = 0; $i < count($messages); $i++) {
            $entries[] = [
                'Id' => 'unique_is_msg' . $i, // REQUIRED
                'ReceiptHandle' => $messages[$i]['ReceiptHandle'], // REQUIRED
                'VisibilityTimeout' => 3600
            ];
```

```
        }
        $result = $client->changeMessageVisibilityBatch([
            'Entries' => $entries,
            'QueueUrl' => $queueUrl
        ]);

        var_dump($result);
    } else {
        echo "No messages in queue \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Mengirim dan menerima pesan di Amazon SQS dengan AWS SDK for PHP Versi 3

Untuk mempelajari tentang pesan Amazon SQS, lihat Mengirim Pesan ke Antrian SQS dan Menerima serta Menghapus Pesan dari Antrian SQS di Panduan Pengguna Service Quotas.

Contoh berikut menunjukkan cara:

- Mengirimkan pesan ke antrian tertentu menggunakan SendMessage.
- Ambil satu atau beberapa pesan (hingga 10) dari antrian tertentu menggunakan. ReceiveMessage
- Menghapus pesan dari antrian menggunakan DeleteMessage.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Kirim pesan

Impor

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

```
use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

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

Menerima dan menghapus pesan

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```
$queueUrl = "QUEUE_URL";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);

try {
    $result = $client->receiveMessage([
        'AttributeNames' => ['SentTimestamp'],
        'MaxNumberOfMessages' => 1,
        'MessageAttributeNames' => ['All'],
        'QueueUrl' => $queueUrl, // REQUIRED
        'WaitTimeSeconds' => 0,
    ]);
    if (!empty($result->get('Messages'))) {
        var_dump($result->get('Messages')[0]);
        $result = $client->deleteMessage([
            'QueueUrl' => $queueUrl, // REQUIRED
            'ReceiptHandle' => $result->get('Messages')[0]['ReceiptHandle'] // REQUIRED
        ]);
    } else {
        echo "No messages in queue. \n";
    }
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Menggunakan antrian huruf mati di Amazon SQS dengan Versi 3 AWS SDK for PHP

Antrian huruf mati adalah antrian (sumber) lain yang dapat ditargetkan untuk pesan yang tidak dapat diproses dengan sukses. Anda dapat menyisihkan dan mengisolasi pesan-pesan ini dalam antrian huruf mati untuk menentukan mengapa pemrosesan mereka tidak berhasil. Anda harus mengkonfigurasi secara individual setiap antrian sumber yang mengirim pesan ke antrian huruf mati. Beberapa antrian dapat menargetkan antrian huruf mati tunggal.

Untuk mempelajari lebih lanjut, lihat Menggunakan Antrian Surat Mati SQS.

Contoh berikut menunjukkan cara:

- Aktifkan antrian huruf mati menggunakan. SetQueueAttributes

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Aktifkan antrian huruf mati

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

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

Menggunakan antrian di Amazon AWS SDK for PHP SQS dengan Versi 3

Untuk mempelajari antrian Amazon SQS, lihat Cara Kerja Antrian SQS.

Contoh berikut menunjukkan cara:

- Kembalikan daftar antrian Anda menggunakan. ListQueues
- Buat antrian baru menggunakan CreateQueue.
- Kembalikan URL antrian yang ada menggunakan GetQueueUrl.
- Hapus antrian tertentu menggunakan DeleteQueue.

Semua kode contoh untuk AWS SDK for PHP tersedia di sini GitHub.

Kredensial

Sebelum menjalankan kode contoh, konfigurasikan AWS kredenal Anda, seperti yang dijelaskan dalam. the section called “Autentikasi dengan AWS” Kemudian impor AWS SDK for PHP, seperti yang dijelaskan dalamthe section called “Menginstal ”.

Kembalikan daftar antrian

Impor

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

use Aws\Exception\AwsException;
```

```
use Aws\Sqs\SqsClient;
```

Kode Sampel

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

Membuat antrean

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

```

$queueName = "SQS_QUEUE_NAME";

$client = new SqsClient([
    'profile' => 'default',
    'region' => 'us-west-2',
    'version' => '2012-11-05'
]);
```

```
try {
    $result = $client->createQueue([
        'QueueName' => $queueName,
        'Attributes' => [
            'DelaySeconds' => 5,
            'MaximumMessageSize' => 4096, // 4 KB
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Kembalikan URL antrian

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

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

Hapus antrian

Impor

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Kode Sampel

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

Kirim acara ke titik akhir EventBridge global Amazon menggunakan AWS SDK for PHP Versi 3

Anda dapat menggunakan titik akhir EventBridge global Amazon untuk meningkatkan ketersediaan dan keandalan aplikasi berbasis peristiwa.

Setelah endpoint EventBridge global diatur, Anda dapat mengirim acara ke sana dengan menggunakan SDK for PHP.

> **Important**
>
> Untuk menggunakan endpoint EventBridge global dengan SDK for PHP, lingkungan PHP Anda harus memiliki AWS ekstensi Common Runtime AWS (CRT) yang diinstal.

Contoh berikut menggunakan PutEventsmetode `EventBridgeClient` untuk mengirim peristiwa tunggal ke endpoint EventBridge global.

```
<?php
/* Send a single event to an existing Amazon EventBridge global endpoint. */
require '../vendor/autoload.php';

use Aws\EventBridge\EventBridgeClient;

$evClient = new EventBridgeClient([
    'region' => 'us-east-1'
]);

$endpointId = 'xxxx123456.xxx';  // Existing EventBridge global endpointId.
$eventBusName = 'default';       // Existing event bus in the us-east-1 Region.

$event =  [
    'Source' => 'my-php-app',
    'DetailType' => 'test',
    'Detail' => json_encode(['foo' => 'bar']),
    'Time' => new DateTime(),
    'Resources' => ['php-script'],
    'EventBusName' => $eventBusName,
    'TraceHeader' => 'test'
];

$result = $evClient->putEvents([
    'EndpointId' => $endpointId,
    'Entries' => [$event]
]);
```

Posting blog ini berisi informasi lebih lanjut tentang titik akhir EventBridge global.

Contoh kode SDK for PHP

Contoh kode dalam topik ini menunjukkan cara menggunakan AWS SDK for PHP with AWS.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Beberapa layanan berisi kategori contoh tambahan yang menunjukkan cara memanfaatkan pustaka atau fungsi khusus untuk layanan.

Layanan

- Contoh API Gateway menggunakan SDK for PHP
- Contoh Aurora menggunakan SDK for PHP
- Contoh Auto Scaling menggunakan SDK for PHP
- Contoh Amazon Bedrock menggunakan SDK for PHP
- Contoh Amazon Bedrock Runtime menggunakan SDK for PHP
- Contoh Amazon DocumentDB menggunakan SDK for PHP
- Contoh DynamoDB menggunakan SDK for PHP
- EC2 Contoh Amazon menggunakan SDK for PHP
- AWS Glue contoh menggunakan SDK for PHP
- Contoh IAM menggunakan SDK for PHP
- Contoh Kinesis menggunakan SDK for PHP
- AWS KMS contoh menggunakan SDK for PHP
- Contoh Lambda menggunakan SDK for PHP
- Contoh MSK Amazon menggunakan SDK for PHP

- Contoh Amazon RDS menggunakan SDK for PHP
- Contoh Layanan Data Amazon RDS menggunakan SDK for PHP
- Contoh Amazon Rekognition menggunakan SDK for PHP
- Contoh Amazon S3 menggunakan SDK for PHP
- Contoh Bucket Direktori S3 menggunakan SDK for PHP
- Amazon SES contoh menggunakan SDK for PHP
- Contoh Amazon SNS menggunakan SDK for PHP
- Contoh Amazon SQS menggunakan SDK for PHP

Contoh API Gateway menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan API Gateway AWS SDK for PHP with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan
- Skenario

Tindakan

`GetBasePathMapping`

Contoh kode berikut menunjukkan cara menggunakan`GetBasePathMapping`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
```

```
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- Untuk detail API, lihat GetBasePathMappingdi Referensi AWS SDK for PHP API.

`ListBasePathMappings`

Contoh kode berikut menunjukkan cara menggunakan`ListBasePathMappings`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* //////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
```

```
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- Untuk detail API, lihat ListBasePathMappingsdi Referensi AWS SDK for PHP API.

`UpdateBasePathMapping`

Contoh kode berikut menunjukkan cara menggunakan`UpdateBasePathMapping`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * //////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
```

```
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- Untuk detail API, lihat UpdateBasePathMappingdi Referensi AWS SDK for PHP API.

Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

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

Contoh Aurora menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP with Aurora.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Skenario

Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

SDK untuk PHP

Menunjukkan cara menggunakan AWS SDK for PHP untuk membuat aplikasi web yang melacak item pekerjaan dalam database Amazon RDS dan laporan email dengan menggunakan Amazon Simple Email Service (Amazon SES). Contoh ini menggunakan front end yang dibangun dengan React.js untuk berinteraksi dengan backend RESTful PHP.

- Integrasikan aplikasi web React.js dengan AWS layanan.
- Cantumkan, tambahkan, perbarui, dan hapus butir di tabel Amazon RDS.
- Kirim laporan email tentang butir kerja terfilter dengan menggunakan Amazon SES.
- Menyebarkan dan mengelola sumber daya contoh dengan AWS CloudFormation skrip yang disertakan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

Contoh Auto Scaling menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Auto Scaling AWS SDK for PHP with.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan

Memulai

Halo Auto Scaling

Contoh kode berikut menunjukkan cara memulai menggunakan Auto Scaling.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- Untuk detail API, lihat DescribeAutoScalingGroupsdi Referensi AWS SDK for PHP API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat grup EC2 Auto Scaling Amazon dengan template peluncuran dan Availability Zone, dan dapatkan informasi tentang menjalankan instans.
- Aktifkan pengumpulan CloudWatch metrik Amazon.
- Perbarui kapasitas yang diinginkan grup dan tunggu instance dimulai.
- Mengakhiri sebuah instance dalam grup.
- Buat daftar aktivitas penskalaan yang terjadi sebagai respons terhadap permintaan pengguna dan perubahan kapasitas.
- Dapatkan statistik untuk CloudWatch metrik, lalu bersihkan sumber daya.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
```

```
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
```

```
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");
```

```
        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
```

```
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
```

```
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
```

```
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }
```

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Tindakan

`CreateAutoScalingGroup`

Contoh kode berikut menunjukkan cara menggunakan`CreateAutoScalingGroup`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Untuk detail API, lihat CreateAutoScalingGroupdi Referensi AWS SDK for PHP API.

`DeleteAutoScalingGroup`

Contoh kode berikut menunjukkan cara menggunakan`DeleteAutoScalingGroup`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Untuk detail API, lihat DeleteAutoScalingGroupdi Referensi AWS SDK for PHP API.

`DescribeAutoScalingGroups`

Contoh kode berikut menunjukkan cara menggunakan`DescribeAutoScalingGroups`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- Untuk detail API, lihat DescribeAutoScalingGroupsdi Referensi AWS SDK for PHP API.

`DescribeAutoScalingInstances`

Contoh kode berikut menunjukkan cara menggunakan`DescribeAutoScalingInstances`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Untuk detail API, lihat DescribeAutoScalingInstancesdi Referensi AWS SDK for PHP API.

`DescribeScalingActivities`

Contoh kode berikut menunjukkan cara menggunakan`DescribeScalingActivities`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Untuk detail API, lihat DescribeScalingActivitiesdi Referensi AWS SDK for PHP API.

`DisableMetricsCollection`

Contoh kode berikut menunjukkan cara menggunakan`DisableMetricsCollection`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Untuk detail API, lihat DisableMetricsCollectiondi Referensi AWS SDK for PHP API.

`EnableMetricsCollection`

Contoh kode berikut menunjukkan cara menggunakan`EnableMetricsCollection`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
```

```
    }
```

- Untuk detail API, lihat EnableMetricsCollectiondi Referensi AWS SDK for PHP API.

`SetDesiredCapacity`

Contoh kode berikut menunjukkan cara menggunakan`SetDesiredCapacity`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- Untuk detail API, lihat SetDesiredCapacitydi Referensi AWS SDK for PHP API.

`TerminateInstanceInAutoScalingGroup`

Contoh kode berikut menunjukkan cara menggunakan`TerminateInstanceInAutoScalingGroup`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient->terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' => $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
                error_log("Cannot terminate an instance while it is still pending.
Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- Untuk detail API, lihat TerminateInstanceInAutoScalingGroupdi Referensi AWS SDK for PHP API.

`UpdateAutoScalingGroup`

Contoh kode berikut menunjukkan cara menggunakan`UpdateAutoScalingGroup`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- Untuk detail API, lihat UpdateAutoScalingGroupdi Referensi AWS SDK for PHP API.

Contoh Amazon Bedrock menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP With Amazon Bedrock.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan

Tindakan

`ListFoundationModels`

Contoh kode berikut menunjukkan cara menggunakan`ListFoundationModels`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat daftar model foundation Amazon Bedrock yang tersedia.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- Untuk detail API, lihat ListFoundationModelsdi Referensi AWS SDK for PHP API.

Contoh Amazon Bedrock Runtime menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Runtime AWS SDK for PHP with Amazon Bedrock.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Skenario
- Amazon Nova

- Generator Gambar Amazon Titan
- Antropik Claude
- Difusi Stabil

Skenario

Gunakan beberapa model fondasi di Amazon Bedrock

Contoh kode berikut menunjukkan cara menyiapkan dan mengirim prompt ke berbagai model berbahasa besar (LLMs) di Amazon Bedrock

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Panggil beberapa LLMs di Amazon Bedrock.

```
namespace BedrockRuntime;

class GettingStartedWithBedrockRuntime
{
    protected BedrockRuntimeService $bedrockRuntimeService;
    public function runExample()
    {
        echo "\n";
        echo "---------------------------------------------------------------------
\n";
        echo "Welcome to the Amazon Bedrock Runtime getting started demo using PHP!
\n";
        echo "---------------------------------------------------------------------
\n";
        $bedrockRuntimeService = new BedrockRuntimeService();
        $prompt = 'In one paragraph, who are you?';
        echo "\nPrompt: " . $prompt;
        echo "\n\nAnthropic Claude:\n";
        echo $bedrockRuntimeService->invokeClaude($prompt);
```

```
        echo
 "\n----------------------------------------------------------------------\n";
        $image_prompt = 'stylized picture of a cute old steampunk robot';
        echo "\nImage prompt: " . $image_prompt;
        echo "\n\nStability.ai Stable Diffusion XL:\n";
        $diffusionSeed = rand(0, 4294967295);
        $style_preset = 'photographic';
        $base64 = $bedrockRuntimeService->invokeStableDiffusion($image_prompt,
 $diffusionSeed, $style_preset);
        $image_path = $this->saveImage($base64, 'stability.stable-diffusion-xl');
        echo "The generated image has been saved to $image_path";
        echo "\n\nAmazon Titan Image Generation:\n";
        $titanSeed = rand(0, 2147483647);
        $base64 = $bedrockRuntimeService->invokeTitanImage($image_prompt,
 $titanSeed);
        $image_path = $this->saveImage($base64, 'amazon.titan-image-generator-v2');
        echo "The generated image has been saved to $image_path";
    }

    private function saveImage($base64_image_data, $model_id): string
    {
        $output_dir = "output";
        if (!file_exists($output_dir)) {
            mkdir($output_dir);
        }

        $i = 1;
        while (file_exists("$output_dir/$model_id" . '_' . "$i.png")) {
            $i++;
        }

        $image_data = base64_decode($base64_image_data);
        $file_path = "$output_dir/$model_id" . '_' . "$i.png";
        $file = fopen($file_path, 'wb');
        fwrite($file, $image_data);
        fclose($file);
        return $file_path;
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - InvokeModel

- InvokeModelWithResponseStream

Amazon Nova

Bercakap-cakap

Contoh kode berikut menunjukkan cara mengirim pesan teks ke Amazon Nova, menggunakan API Converse Bedrock.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Kirim pesan teks ke Amazon Nova, menggunakan API Converse Bedrock.

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

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

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

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

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
```

```
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

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

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

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

- Untuk detail API, lihat Converse di Referensi AWS SDK for PHP API.

Generator Gambar Amazon Titan

InvokeModel

Contoh kode berikut menunjukkan cara memanggil Amazon Titan Image di Amazon Bedrock untuk menghasilkan gambar.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat gambar dengan Amazon Titan Image Generator.

```
    public function invokeTitanImage(string $prompt, int $seed)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and default values for Titan Image models refer
 to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html

        $base64_image_data = "";
        try {
            $modelId = 'amazon.titan-image-generator-v2:0';
            $request = json_encode([
                'taskType' => 'TEXT_IMAGE',
                'textToImageParams' => [
                    'text' => $prompt
                ],
                'imageGenerationConfig' => [
                    'numberOfImages' => 1,
                    'quality' => 'standard',
                    'cfgScale' => 8.0,
                    'height' => 512,
                    'width' => 512,
                    'seed' => $seed
                ]
```

```
            ]);
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => $request,
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for PHP API.

Antropik Claude

InvokeModel

Contoh kode berikut menunjukkan cara mengirim pesan teks ke Anthropic Claude, menggunakan Invoke Model API.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Gunakan model dasar Anthropic Claude 2 untuk menghasilkan teks.

```
    public function invokeClaude($prompt)
    {
        // The different model providers have individual request and response
formats.
        // For the format, ranges, and default values for Anthropic Claude, refer
to:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
claude.html

        $completion = "";
        try {
            $modelId = 'anthropic.claude-3-haiku-20240307-v1:0';
        // Claude requires you to enclose the prompt as follows:
            $body = [
                'anthropic_version' => 'bedrock-2023-05-31',
                'max_tokens' => 512,
                'temperature' => 0.5,
                'messages' => [[
                    'role' => 'user',
                    'content' => $prompt
                ]]
            ];
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $completion = $response_body->content[0]->text;
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $completion;
    }
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for PHP API.

Difusi Stabil

InvokeModel

Contoh kode berikut menunjukkan cara memanggil Stability.ai Stable Diffusion XL di Amazon Bedrock untuk menghasilkan gambar.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat gambar dengan Difusi Stabil.

```
    public function invokeStableDiffusion(string $prompt, int $seed, string
 $style_preset)
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available style_presets of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-diffusion-xl-v1';
            $body = [
                'text_prompts' => [
                    ['text' => $prompt]
                ],
                'seed' => $seed,
                'cfg_scale' => 10,
                'steps' => 30
            ];
            if ($style_preset) {
                $body['style_preset'] = $style_preset;
            }

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->artifacts[0]->base64;
        } catch (Exception $e) {
```

```
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for PHP API.

Contoh Amazon DocumentDB menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Amazon DocumentDB. AWS SDK for PHP

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon DocumentDB

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari aliran perubahan DocumentDB. Fungsi mengambil payload DocumentDB dan mencatat isi catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Amazon DocumentDB dengan Lambda menggunakan PHP.

```
<?php
```

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Contoh DynamoDB menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP with DynamoDB.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat tabel yang dapat menyimpan data film.
- Masukkan, dapatkan, dan perbarui satu film dalam tabel tersebut.
- Tulis data film ke tabel dari file JSON sampel.
- Kueri untuk film yang dirilis pada tahun tertentu.
- Pindai film yang dirilis dalam suatu rentang tahun.
- Hapus film dari tabel, lalu hapus tabel tersebut.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Karena contoh ini menggunakan file pendukung, pastikan untuk membaca panduan dalam contoh PHP README.md file.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );
```

```
        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
```

```
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";
```

```
        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }
```

```
        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Kueri
  - Scan
  - UpdateItem

Tindakan

`BatchExecuteStatement`

Contoh kode berikut menunjukkan cara menggunakan`BatchExecuteStatement`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
```

```
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
```

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

- Untuk detail API, lihat BatchExecuteStatementdi Referensi AWS SDK for PHP API.

`BatchWriteItem`

Contoh kode berikut menunjukkan cara menggunakan`BatchWriteItem`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
        }

        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
```

```
                echo $e->getMessage();
                die();
            }
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.\n";
                break;
            }
        }
    }
```

- Untuk detail API, lihat BatchWriteItemdi Referensi AWS SDK for PHP API.

`CreateTable`

Contoh kode berikut menunjukkan cara menggunakan`CreateTable`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat tabel.

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
```

```
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- Untuk detail API, lihat CreateTabledi Referensi AWS SDK for PHP API.

`DeleteItem`

Contoh kode berikut menunjukkan cara menggunakan`DeleteItem`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
```

```
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Untuk detail API, lihat DeleteItemdi Referensi AWS SDK for PHP API.

`DeleteTable`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTable`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
            ]);
        });
    }
```

- Untuk detail API, lihat DeleteTabledi Referensi AWS SDK for PHP API.

`ExecuteStatement`

Contoh kode berikut menunjukkan cara menggunakan`ExecuteStatement`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
```

```
            'Parameters' => $parameters,
        ]);
    }
```

- Untuk detail API, lihat ExecuteStatementdi Referensi AWS SDK for PHP API.

`GetItem`

Contoh kode berikut menunjukkan cara menggunakan`GetItem`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Untuk detail API, lihat GetItemdi Referensi AWS SDK for PHP API.

`ListTables`

Contoh kode berikut menunjukkan cara menggunakan`ListTables`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- Untuk detail API, lihat ListTablesdi Referensi AWS SDK for PHP API.

`PutItem`

Contoh kode berikut menunjukkan cara menggunakan`PutItem`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
    }
```

```
        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- Untuk detail API, lihat PutItemdi Referensi AWS SDK for PHP API.

`Query`

Contoh kode berikut menunjukkan cara menggunakan`Query`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
```

```
    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- Untuk detail API, lihat Kueri di Referensi API AWS SDK for PHP .

`Scan`

Contoh kode berikut menunjukkan cara menggunakan`Scan`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $yearsKey = [
```

```
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- Untuk detail API, lihat Scan di Referensi API AWS SDK for PHP .

`UpdateItem`

Contoh kode berikut menunjukkan cara menggunakan`UpdateItem`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- Untuk detail API, lihat UpdateItemdi Referensi AWS SDK for PHP API.

Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

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

Melakukan kueri pada tabel menggunakan batch pernyataan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Dapatkan batch item dengan menjalankan beberapa pernyataan SELECT.
- Tambahkan batch item dengan menjalankan beberapa pernyataan INSERT.
- Perbarui batch item dengan menjalankan beberapa pernyataan UPDATE.
- Hapus batch item dengan menjalankan beberapa pernyataan DELETE.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
```

```
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";
```

```
        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
```

```
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
```

```
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Untuk detail API, lihat BatchExecuteStatementdi Referensi AWS SDK for PHP API.

Melakukan kueri tabel menggunakan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Dapatkan item dengan menjalankan pernyataan SELECT.
- Tambahkan item dengan menjalankan pernyataan INSERT.
- Perbarui item dengan menjalankan pernyataan UPDATE.
- Hapus item dengan menjalankan pernyataan DELETE.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");
```

```
        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
```

```
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";
```

```
        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
```

```
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- Untuk detail API, lihat ExecuteStatementdi Referensi AWS SDK for PHP API.

Contoh nirserver

Memanggil fungsi Lambda dari pemicu DynamoDB

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran DynamoDB. Fungsi mengambil payload DynamoDB dan mencatat isi catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara DynamoDB dengan Lambda menggunakan PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

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

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu DynamoDB

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran DynamoDB. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch DynamoDB dengan Lambda menggunakan PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
```

```
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

EC2 Contoh Amazon menggunakan SDK for PHP

Contoh kode berikut menunjukkan kepada Anda cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP With Amazon EC2.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan

Tindakan

`CreateVpc`

Contoh kode berikut menunjukkan cara menggunakan`CreateVpc`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat CreateVpcdi Referensi AWS SDK for PHP API.

`CreateVpcEndpoint`

Contoh kode berikut menunjukkan cara menggunakan`CreateVpcEndpoint`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat CreateVpcEndpointdi Referensi AWS SDK for PHP API.

`DeleteVpc`

Contoh kode berikut menunjukkan cara menggunakan`DeleteVpc`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat DeleteVpcdi Referensi AWS SDK for PHP API.

`DeleteVpcEndpoints`

Contoh kode berikut menunjukkan cara menggunakan`DeleteVpcEndpoints`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat DeleteVpcEndpointsdi Referensi AWS SDK for PHP API.

`DescribeRouteTables`

Contoh kode berikut menunjukkan cara menggunakan`DescribeRouteTables`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * @param array $routeTableIds
     * @param array $filters
     * @return array
     */
```

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

- Untuk detail API, lihat DescribeRouteTablesdi Referensi AWS SDK for PHP API.

AWS Glue contoh menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP with AWS Glue.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar
- Tindakan

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat crawler yang merayapi bucket Amazon S3 publik dan membuat database metadata berformat CSV.
- Buat daftar informasi tentang database dan tabel di situs Anda AWS Glue Data Catalog.
- Buat pekerjaan untuk mengekstrak data CSV dari bucket S3, mengubah data, dan memuat output berformat JSON ke bucket S3 lain.
- Buat daftar informasi tentang menjalankan pekerjaan, melihat data yang diubah, dan membersihkan sumber daya.

Untuk informasi selengkapnya, lihat Tutorial: Memulai AWS Glue Studio.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;
```

```
class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
 $databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";
```

```
        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];
```

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

echo "Downloading " . $objects[1]['Key'] . "\n";
/** @var Stream $downloadObject */
$downloadObject = $s3client->getObject([
    'Bucket' => $bucketName,
    'Key' => $objects[1]['Key'],
])['Body']->getContents();
echo "Here is the first 1000 characters in the object.";
echo substr($downloadObject, 0, 1000);

$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

echo "Delete the databases.\n";
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);

$deleteObjects = $s3client->listObjectsV2([
    'Bucket' => $bucketName,
]);
echo "Delete all objects in the bucket.\n";
```

```
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
```

```
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
```

```
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
```

```
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

```
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Tindakan

`CreateCrawler`

Contoh kode berikut menunjukkan cara menggunakan`CreateCrawler`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Untuk detail API, lihat CreateCrawlerdi Referensi AWS SDK for PHP API.

`CreateJob`

Contoh kode berikut menunjukkan cara menggunakan`CreateJob`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- Untuk detail API, lihat CreateJobdi Referensi AWS SDK for PHP API.

`DeleteCrawler`

Contoh kode berikut menunjukkan cara menggunakan`DeleteCrawler`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
```

```
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Untuk detail API, lihat DeleteCrawlerdi Referensi AWS SDK for PHP API.

`DeleteDatabase`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDatabase`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Untuk detail API, lihat DeleteDatabasedi Referensi AWS SDK for PHP API.

`DeleteJob`

Contoh kode berikut menunjukkan cara menggunakan`DeleteJob`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Untuk detail API, lihat DeleteJobdi Referensi AWS SDK for PHP API.

`DeleteTable`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTable`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "Delete the tables.\n";
```

```
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Untuk detail API, lihat DeleteTabledi Referensi AWS SDK for PHP API.

`GetCrawler`

Contoh kode berikut menunjukkan cara menggunakan`GetCrawler`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
```

```
        });
    }
```

- Untuk detail API, lihat GetCrawlerdi Referensi AWS SDK for PHP API.

`GetDatabase`

Contoh kode berikut menunjukkan cara menggunakan`GetDatabase`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- Untuk detail API, lihat GetDatabasedi Referensi AWS SDK for PHP API.

`GetJobRun`

Contoh kode berikut menunjukkan cara menggunakan`GetJobRun`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Untuk detail API, lihat GetJobRundi Referensi AWS SDK for PHP API.

`GetJobRuns`

Contoh kode berikut menunjukkan cara menggunakan`GetJobRuns`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Untuk detail API, lihat GetJobRunsdi Referensi AWS SDK for PHP API.

`GetTables`

Contoh kode berikut menunjukkan cara menggunakan`GetTables`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $databaseName = "doc-example-database-$uniqid";
```

```
        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Untuk detail API, lihat GetTablesdi Referensi AWS SDK for PHP API.

`ListJobs`

Contoh kode berikut menunjukkan cara menggunakan`ListJobs`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
```

```
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Untuk detail API, lihat ListJobsdi Referensi AWS SDK for PHP API.

`StartCrawler`

Contoh kode berikut menunjukkan cara menggunakan`StartCrawler`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Untuk detail API, lihat StartCrawlerdi Referensi AWS SDK for PHP API.

`StartJobRun`

Contoh kode berikut menunjukkan cara menggunakan`StartJobRun`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Untuk detail API, lihat StartJobRundi Referensi AWS SDK for PHP API.

Contoh IAM menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP with IAM.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar
- Tindakan

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut menunjukkan cara membuat pengguna dan mengambil peran.

> **Warning**
>
> Untuk menghindari risiko keamanan, jangan gunakan pengguna IAM untuk otentikasi saat mengembangkan perangkat lunak yang dibuat khusus atau bekerja dengan data nyata. Sebaliknya, gunakan federasi dengan penyedia identitas seperti AWS IAM Identity Center.

- Buat pengguna tanpa izin.
- Buat peran yang memberikan izin untuk mencantumkan bucket Amazon S3 untuk akun tersebut.
- Tambahkan kebijakan agar pengguna dapat mengambil peran tersebut.
- Asumsikan peran dan daftar bucket S3 menggunakan kredenal sementara, lalu bersihkan sumber daya.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";
```

```
$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
```

```
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Tindakan

`AttachRolePolicy`

Contoh kode berikut menunjukkan cara menggunakan`AttachRolePolicy`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
```

```
    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- Untuk detail API, lihat AttachRolePolicydi Referensi AWS SDK for PHP API.

`CreatePolicy`

Contoh kode berikut menunjukkan cara menggunakan`CreatePolicy`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
```

```
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
{
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- Untuk detail API, lihat CreatePolicydi Referensi AWS SDK for PHP API.

`CreateRole`

Contoh kode berikut menunjukkan cara menggunakan`CreateRole`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
```

```
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- Untuk detail API, lihat CreateRoledi Referensi AWS SDK for PHP API.

`CreateServiceLinkedRole`

Contoh kode berikut menunjukkan cara menggunakan`CreateServiceLinkedRole`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();
```

```
    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- Untuk detail API, lihat CreateServiceLinkedRoledi Referensi AWS SDK for PHP API.

`CreateUser`

Contoh kode berikut menunjukkan cara menggunakan`CreateUser`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
```

```
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- Untuk detail API, lihat CreateUserdi Referensi AWS SDK for PHP API.

`GetAccountPasswordPolicy`

Contoh kode berikut menunjukkan cara menggunakan`GetAccountPasswordPolicy`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Untuk detail API, lihat GetAccountPasswordPolicydi Referensi AWS SDK for PHP API.

`GetPolicy`

Contoh kode berikut menunjukkan cara menggunakan`GetPolicy`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- Untuk detail API, lihat GetPolicydi Referensi AWS SDK for PHP API.

`GetRole`

Contoh kode berikut menunjukkan cara menggunakan`GetRole`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
```

```
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Untuk detail API, lihat GetRoledi Referensi AWS SDK for PHP API.

`ListAttachedRolePolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListAttachedRolePolicies`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- Untuk detail API, lihat ListAttachedRolePoliciesdi Referensi AWS SDK for PHP API.

`ListGroups`

Contoh kode berikut menunjukkan cara menggunakan`ListGroups`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- Untuk detail API, lihat ListGroupsdi Referensi AWS SDK for PHP API.

`ListPolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListPolicies`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- Untuk detail API, lihat ListPoliciesdi Referensi AWS SDK for PHP API.

`ListRolePolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListRolePolicies`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- Untuk detail API, lihat ListRolePoliciesdi Referensi AWS SDK for PHP API.

`ListRoles`

Contoh kode berikut menunjukkan cara menggunakan`ListRoles`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
```

```
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- Untuk detail API, lihat ListRolesdi Referensi AWS SDK for PHP API.

`ListSAMLProviders`

Contoh kode berikut menunjukkan cara menggunakan`ListSAMLProviders`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Untuk detail API, lihat Daftar SAMLProviders di Referensi AWS SDK for PHP API.

`ListUsers`

Contoh kode berikut menunjukkan cara menggunakan`ListUsers`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- Untuk detail API, lihat ListUsersdi Referensi AWS SDK for PHP API.

Contoh Kinesis menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP Kinesis with.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Kinesis dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
```

```
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

AWS KMS contoh menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP with AWS KMS.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan

Memulai

Halo AWS KMS

Contoh kode berikut menunjukkan bagaimana untuk mulai menggunakan AWS Key Management Service.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\Kms\KmsClient;

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

$client = new KmsClient([]);

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

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

- Untuk detail API, lihat ListKeysdi Referensi AWS SDK for PHP API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat kunci KMS.
- Buat daftar kunci KMS untuk akun Anda dan dapatkan detailnya.
- Aktifkan dan nonaktifkan tombol KMS.
- Hasilkan kunci data simetris yang dapat digunakan untuk enkripsi sisi klien.
- Hasilkan kunci asimetris yang digunakan untuk menandatangani data secara digital.
- Tombol tag.
- Hapus kunci KMS.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

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

This KMS Basics scenario creates two key types:
```

```
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

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

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

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

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

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

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

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

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

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
```

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

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

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

```
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

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

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

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
```

```
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

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

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

```
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

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

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
changes to PendingDeletion and the key can't be used in any cryptographic
operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

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

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

```
            $this->cleanUp();
        }

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



namespace Kms;

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

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

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


    /***
     * @param string $keySpec
```

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



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
"SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
```

```
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
```

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



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

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



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



    /***
     * @return array
```

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



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



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

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


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



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
```

```
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
```

```
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



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



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
```

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



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



    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
```

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


}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Dekripsi
  - DescribeKey
  - DisableKey
  - EnableKey
  - Enkripsi
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Tanda
  - TagResource

Tindakan

`CreateAlias`

Contoh kode berikut menunjukkan cara menggunakan`CreateAlias`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat CreateAliasdi Referensi AWS SDK for PHP API.

`CreateGrant`

Contoh kode berikut menunjukkan cara menggunakan`CreateGrant`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat CreateGrantdi Referensi AWS SDK for PHP API.

`CreateKey`

Contoh kode berikut menunjukkan cara menggunakan`CreateKey`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

```
        }
    }
```

- Untuk detail API, lihat CreateKeydi Referensi AWS SDK for PHP API.

`Decrypt`

Contoh kode berikut menunjukkan cara menggunakan`Decrypt`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Mendekripsi di Referensi AWS SDK for PHP API.

`DeleteAlias`

Contoh kode berikut menunjukkan cara menggunakan`DeleteAlias`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat DeleteAliasdi Referensi AWS SDK for PHP API.

`DescribeKey`

Contoh kode berikut menunjukkan cara menggunakan`DescribeKey`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat DescribeKeydi Referensi AWS SDK for PHP API.

`DisableKey`

Contoh kode berikut menunjukkan cara menggunakan`DisableKey`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat DisableKeydi Referensi AWS SDK for PHP API.

`EnableKey`

Contoh kode berikut menunjukkan cara menggunakan`EnableKey`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat EnableKeydi Referensi AWS SDK for PHP API.

`Encrypt`

Contoh kode berikut menunjukkan cara menggunakan`Encrypt`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * @param string $keyId
     * @param string $text
     * @return Result
```

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

- Untuk detail API, lihat Enkripsi di Referensi AWS SDK for PHP API.

`ListAliases`

Contoh kode berikut menunjukkan cara menggunakan`ListAliases`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
```

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

- Untuk detail API, lihat ListAliasesdi Referensi AWS SDK for PHP API.

`ListGrants`

Contoh kode berikut menunjukkan cara menggunakan`ListGrants`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
```

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

- Untuk detail API, lihat ListGrantsdi Referensi AWS SDK for PHP API.

`ListKeys`

Contoh kode berikut menunjukkan cara menggunakan`ListKeys`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
```

```
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Untuk detail API, lihat ListKeysdi Referensi AWS SDK for PHP API.

`PutKeyPolicy`

Contoh kode berikut menunjukkan cara menggunakan`PutKeyPolicy`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

```
        }
    }
```

- Untuk detail API, lihat PutKeyPolicydi Referensi AWS SDK for PHP API.

`RevokeGrant`

Contoh kode berikut menunjukkan cara menggunakan`RevokeGrant`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat RevokeGrantdi Referensi AWS SDK for PHP API.

`ScheduleKeyDeletion`

Contoh kode berikut menunjukkan cara menggunakan`ScheduleKeyDeletion`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat ScheduleKeyDeletiondi Referensi AWS SDK for PHP API.

`Sign`

Contoh kode berikut menunjukkan cara menggunakan`Sign`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat Referensi AWS SDK for PHP API Masuk.

`TagResource`

Contoh kode berikut menunjukkan cara menggunakan`TagResource`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

- Untuk detail API, lihat TagResourcedi Referensi AWS SDK for PHP API.

Contoh Lambda menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Lambda AWS SDK for PHP with.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat peran IAM dan fungsi Lambda, lalu unggah kode handler.
- Panggil fungsi dengan satu parameter dan dapatkan hasil.
- Perbarui kode fungsi dan konfigurasikan dengan variabel lingkungan.
- Panggil fungsi dengan parameter baru dan dapatkan hasil. Tampilkan log eksekusi yang dikembalikan.
- Buat daftar fungsi untuk akun Anda, lalu bersihkan sumber daya.

Untuk informasi selengkapnya, lihat Membuat fungsi Lambda dengan konsol.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
```

```
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";

        echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
        $bucketName = "amzn-s3-demo-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName, $role,
 $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);
```

```
        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
```

```
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
$result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
```

```
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Memohon
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Tindakan

`CreateFunction`

Contoh kode berikut menunjukkan cara menggunakan`CreateFunction`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- Untuk detail API, lihat CreateFunctiondi Referensi AWS SDK for PHP API.

`DeleteFunction`

Contoh kode berikut menunjukkan cara menggunakan`DeleteFunction`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Untuk detail API, lihat DeleteFunctiondi Referensi AWS SDK for PHP API.

`GetFunction`

Contoh kode berikut menunjukkan cara menggunakan`GetFunction`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Untuk detail API, lihat GetFunctiondi Referensi AWS SDK for PHP API.

`Invoke`

Contoh kode berikut menunjukkan cara menggunakan`Invoke`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- Untuk detail API, lihat Memanggil di Referensi AWS SDK for PHP API.

`ListFunctions`

Contoh kode berikut menunjukkan cara menggunakan`ListFunctions`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
public function listFunctions($maxItems = 50, $marker = null)
{
    if (is_null($marker)) {
        return $this->lambdaClient->listFunctions([
            'MaxItems' => $maxItems,
        ]);
    }
```

```
        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- Untuk detail API, lihat ListFunctionsdi Referensi AWS SDK for PHP API.

`UpdateFunctionCode`

Contoh kode berikut menunjukkan cara menggunakan`UpdateFunctionCode`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- Untuk detail API, lihat UpdateFunctionCodedi Referensi AWS SDK for PHP API.

`UpdateFunctionConfiguration`

Contoh kode berikut menunjukkan cara menggunakan`UpdateFunctionConfiguration`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- Untuk detail API, lihat UpdateFunctionConfigurationdi Referensi AWS SDK for PHP API.

Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- API Gateway

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

Contoh nirserver

Menghubungkan ke database Amazon RDS dalam fungsi Lambda

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menghubungkan ke database RDS. Fungsi membuat permintaan database sederhana dan mengembalikan hasilnya.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menghubungkan ke database Amazon RDS dalam fungsi Lambda menggunakan PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
```

```
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
```

```
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

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

Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Kinesis dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data
```

```
            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

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

Memanggil fungsi Lambda dari pemicu DynamoDB

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran DynamoDB. Fungsi mengambil payload DynamoDB dan mencatat isi catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara DynamoDB dengan Lambda menggunakan PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
```

```
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

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

Memanggil fungsi Lambda dari pemicu Amazon DocumentDB

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari aliran perubahan DocumentDB. Fungsi mengambil payload DocumentDB dan mencatat isi catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Amazon DocumentDB dengan Lambda menggunakan PHP.

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
```

```
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Memanggil fungsi Lambda dari pemicu MSK Amazon

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari kluster MSK Amazon. Fungsi mengambil muatan MSK dan mencatat konten catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara MSK Amazon dengan Lambda menggunakan PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

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

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

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

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

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

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara S3 dengan Lambda menggunakan PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();
```

```
        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

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

Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SNS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SQS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
```

```
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

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

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}
```

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

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu DynamoDB

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran DynamoDB. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch DynamoDB dengan Lambda menggunakan PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
```

```
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

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

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch SQS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
```

```
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

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

Contoh MSK Amazon menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan MSK AWS SDK for PHP with Amazon.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu MSK Amazon

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari kluster MSK Amazon. Fungsi mengambil muatan MSK dan mencatat konten catatan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara MSK Amazon dengan Lambda menggunakan PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
```

```
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

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

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

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

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

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

Contoh Amazon RDS menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP With Amazon RDS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan
- Skenario
- Contoh nirserver

Tindakan

`CreateDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBInstance`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
```

```
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Untuk detail API, lihat Membuat DBInstance di Referensi AWS SDK for PHP API.

`CreateDBSnapshot`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBSnapshot`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\Exception\AwsException;
```

```
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Untuk detail API, lihat Membuat DBSnapshot di Referensi AWS SDK for PHP API.

`DeleteDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBInstance`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\Exception\AwsException;


//Create an RDSClient
```

```
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Untuk detail API, lihat Menghapus DBInstance di Referensi AWS SDK for PHP API.

`DescribeDBInstances`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBInstances`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);
```

```
try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Untuk detail API, lihat Menjelaskan DBInstances di Referensi AWS SDK for PHP API.

Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

SDK untuk PHP

Menunjukkan cara menggunakan AWS SDK for PHP untuk membuat aplikasi web yang melacak item pekerjaan dalam database Amazon RDS dan laporan email dengan menggunakan Amazon Simple Email Service (Amazon SES). Contoh ini menggunakan front end yang dibangun dengan React.js untuk berinteraksi dengan backend RESTful PHP.

- Integrasikan aplikasi web React.js dengan AWS layanan.
- Cantumkan, tambahkan, perbarui, dan hapus butir di tabel Amazon RDS.
- Kirim laporan email tentang butir kerja terfilter dengan menggunakan Amazon SES.
- Menyebarkan dan mengelola sumber daya contoh dengan AWS CloudFormation skrip yang disertakan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

Contoh nirserver

Menghubungkan ke database Amazon RDS dalam fungsi Lambda

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menghubungkan ke database RDS. Fungsi membuat permintaan database sederhana dan mengembalikan hasilnya.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menghubungkan ke database Amazon RDS dalam fungsi Lambda menggunakan PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(
```

```
 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
           $connectionConfig['user'],
           $connectionConfig['password'],
           [
               PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
               PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
           ]
       );

       // Obtain the result of the query
       $stmt = $conn->prepare('SELECT ?+? AS sum');
       $stmt->execute([3, 2]);

       return $stmt->fetch(PDO::FETCH_ASSOC);
   }

   /**
    * @param mixed $event
    * @param Context $context
    * @return array
    */
   public function handle(mixed $event, Context $context): array
   {
       $this->logger->info("Processing query");

       // Execute database flow
       $result = $this->getQueryResults();

       return [
           'sum' => $result['sum']
       ];
   }
}

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

Contoh Layanan Data Amazon RDS menggunakan SDK for PHP

Contoh kode berikut menunjukkan kepada Anda cara melakukan tindakan dan mengimplementasikan skenario umum AWS SDK for PHP dengan menggunakan Layanan Data Amazon RDS.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Skenario

Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

SDK untuk PHP

Menunjukkan cara menggunakan AWS SDK for PHP untuk membuat aplikasi web yang melacak item pekerjaan dalam database Amazon RDS dan laporan email dengan menggunakan Amazon Simple Email Service (Amazon SES). Contoh ini menggunakan front end yang dibangun dengan React.js untuk berinteraksi dengan backend RESTful PHP.

- Integrasikan aplikasi web React.js dengan AWS layanan.
- Cantumkan, tambahkan, perbarui, dan hapus butir di tabel Amazon RDS.
- Kirim laporan email tentang butir kerja terfilter dengan menggunakan Amazon SES.
- Menyebarkan dan mengelola sumber daya contoh dengan AWS CloudFormation skrip yang disertakan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS

- Layanan Data Amazon RDS
- Amazon SES

Contoh Amazon Rekognition menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP With Amazon Rekognition.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Skenario

Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Contoh Amazon S3 menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon S3. AWS SDK for PHP

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver

Memulai

Halo Amazon S3

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon S3.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
use Aws\S3\S3Client;

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

- Untuk detail API, lihat ListBucketsdi Referensi AWS SDK for PHP API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat bucket dan mengunggah file ke dalamnya.
- Mengunduh objek dari bucket.
- Menyalin objek ke subfolder di bucket.
- Membuat daftar objek dalam bucket.
- Menghapus objek bucket dan bucket tersebut.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
```

```
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
```

```
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Tindakan

`CopyObject`

Contoh kode berikut menunjukkan cara menggunakan`CopyObject`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Salinan sederhana dari suatu objek.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- Untuk detail API, lihat CopyObjectdi Referensi AWS SDK for PHP API.

`CreateBucket`

Contoh kode berikut menunjukkan cara menggunakan`CreateBucket`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- Untuk detail API, lihat CreateBucketdi Referensi AWS SDK for PHP API.

`DeleteBucket`

Contoh kode berikut menunjukkan cara menggunakan`DeleteBucket`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Hapus bucket kosong.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- Untuk detail API, lihat DeleteBucketdi Referensi AWS SDK for PHP API.

`DeleteObject`

Contoh kode berikut menunjukkan cara menggunakan`DeleteObject`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

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

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

- Untuk detail API, lihat DeleteObjectdi Referensi AWS SDK for PHP API.

`DeleteObjects`

Contoh kode berikut menunjukkan cara menggunakan`DeleteObjects`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Hapus satu set objek dari daftar kunci.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
```

```
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- Untuk detail API, lihat DeleteObjectsdi Referensi AWS SDK for PHP API.

`GetObject`

Contoh kode berikut menunjukkan cara menggunakan`GetObject`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Dapatkan objek.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- Untuk detail API, lihat GetObjectdi Referensi AWS SDK for PHP API.

`ListObjectsV2`

Contoh kode berikut menunjukkan cara menggunakan`ListObjectsV2`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat daftar objek dalam bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- Untuk detail API, lihat ListObjectsV2 di Referensi AWS SDK for PHP API.

`PutObject`

Contoh kode berikut menunjukkan cara menggunakan`PutObject`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Unggah objek ke bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- Untuk detail API, lihat PutObjectdi Referensi AWS SDK for PHP API.

Skenario

Membuat URL yang telah ditetapkan sebelumnya

Contoh kode berikut menunjukkan cara membuat URL presigned untuk Amazon S3 dan mengunggah objek.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the S3
 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in the
 given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
```

```
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();



namespace S3;

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

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

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

```
    }

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

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

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

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


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

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



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



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
```

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



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



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
 $args = [])
```

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



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

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

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



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
```

```
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

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



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

```
        }
    }



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



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }
```

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



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

}
```

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

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

Contoh nirserver

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara S3 dengan Lambda menggunakan PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
```

```
    }
}

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

Contoh Bucket Direktori S3 menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Bucket Direktori AWS SDK for PHP with S3.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Siapkan VPC dan VPC Endpoint.
- Siapkan Kebijakan, Peran, dan Pengguna untuk bekerja dengan bucket direktori S3 dan kelas penyimpanan S3 Express One Zone.
- Buat dua Klien S3.
- Buat dua bucket.
- Buat objek dan salin.
- Menunjukkan perbedaan kinerja.
- Isi ember untuk menunjukkan perbedaan leksikografis.
- Minta pengguna untuk melihat apakah mereka ingin membersihkan sumber daya.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario yang mendemonstrasikan dasar-dasar bucket direktori Amazon S3 dan S3 Express One Zone.

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

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

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

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute resources
 in the same AZ as your Directory
bucket.\n
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program is
 running in an EC2 instance in the same AZ as your Directory buckets will be.\n";
```

```
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a VPC
 Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

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

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

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

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
```

```
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

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

        // 3. Create an additional client using the credentials with S3 Express
 permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3 Express
 permissions.\n";
        echo "This client is created with the credentials associated with the
 user account with the S3 Express policy attached, so it can perform S3 Express
 operations.\n";
        pressEnter();
```

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

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

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
```

```
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

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

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

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

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

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

        // 6. Demonstrate performance difference.
```

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

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

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

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
" nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
```

```
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
 while the directory bucket does not. This is because the normal bucket considers
```

```
 the whole \"key\" to be the object identifies, while the directory bucket actually
 creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

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

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

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
```

```
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
```

```
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
```

```
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
 "MaxKeys" => $max], $args);
```

```
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
```

```
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
 => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }
```

```
   public function deleteObject(string $bucketName, string $fileName, array $args =
[])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
       try {
           $this->client->deleteObject($parameters);
           if ($this->verbose) {
               echo "Deleted the object named: $fileName from $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object deletion before continuing.";
           }
           throw $exception;
       }
   }



   public function listBuckets(array $args = [])
   {
       try {
           $buckets = $this->client->listBuckets($args);
           if ($this->verbose) {
               echo "Retrieved all " . count($buckets) . "\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
               echo "Please fix error with bucket lists before continuing.";
           }
           throw $exception;
       }
       return $buckets;
   }
```

```
    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for PHP .
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObject
- GetObject
- ListObjects
- PutObject

Amazon SES contoh menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for PHP With Amazon SES.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Skenario

Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

SDK untuk PHP

Menunjukkan cara menggunakan AWS SDK for PHP untuk membuat aplikasi web yang melacak item pekerjaan dalam database Amazon RDS dan laporan email dengan menggunakan Amazon Simple Email Service (Amazon SES). Contoh ini menggunakan front end yang dibangun dengan React.js untuk berinteraksi dengan backend RESTful PHP.

- Integrasikan aplikasi web React.js dengan AWS layanan.

- Cantumkan, tambahkan, perbarui, dan hapus butir di tabel Amazon RDS.
- Kirim laporan email tentang butir kerja terfilter dengan menggunakan Amazon SES.
- Menyebarkan dan mengelola sumber daya contoh dengan AWS CloudFormation skrip yang disertakan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

Contoh Amazon SNS menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for PHP dengan Amazon SNS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan
- Skenario
- Contoh nirserver

Tindakan

`CheckIfPhoneNumberIsOptedOut`

Contoh kode berikut menunjukkan cara menggunakan`CheckIfPhoneNumberIsOptedOut`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS SDK for PHP API.

`ConfirmSubscription`

Contoh kode berikut menunjukkan cara menggunakan`ConfirmSubscription`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat ConfirmSubscriptiondi Referensi AWS SDK for PHP API.

`CreateTopic`

Contoh kode berikut menunjukkan cara menggunakan`CreateTopic`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the requested
 region.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat CreateTopicdi Referensi AWS SDK for PHP API.

`DeleteTopic`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTopic`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat DeleteTopicdi Referensi AWS SDK for PHP API.

`GetSMSAttributes`

Contoh kode berikut menunjukkan cara menggunakan`GetSMSAttributes`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat Dapatkan SMSAttributes Referensi AWS SDK for PHP API.

`GetTopicAttributes`

Contoh kode berikut menunjukkan cara menggunakan`GetTopicAttributes`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat GetTopicAttributesdi Referensi AWS SDK for PHP API.

`ListPhoneNumbersOptedOut`

Contoh kode berikut menunjukkan cara menggunakan`ListPhoneNumbersOptedOut`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat ListPhoneNumbersOptedOutdi Referensi AWS SDK for PHP API.

`ListSubscriptions`

Contoh kode berikut menunjukkan cara menggunakan`ListSubscriptions`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat ListSubscriptionsdi Referensi AWS SDK for PHP API.

`ListTopics`

Contoh kode berikut menunjukkan cara menggunakan`ListTopics`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat ListTopicsdi Referensi AWS SDK for PHP API.

`Publish`

Contoh kode berikut menunjukkan cara menggunakan`Publish`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

`SetSMSAttributes`

Contoh kode berikut menunjukkan cara menggunakan`SetSMSAttributes`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat Menyetel SMSAttributes di Referensi AWS SDK for PHP API.

`SetTopicAttributes`

Contoh kode berikut menunjukkan cara menggunakan`SetTopicAttributes`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
```

```
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat SetTopicAttributesdi Referensi AWS SDK for PHP API.

`Subscribe`

Contoh kode berikut menunjukkan cara menggunakan`Subscribe`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Berlangganan alamat email ke suatu topik.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
```

```
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Berlangganan titik akhir HTTP ke suatu topik.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk detail API, lihat Berlangganan di Referensi AWS SDK for PHP API.

`Unsubscribe`

Contoh kode berikut menunjukkan cara menggunakan`Unsubscribe`.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
```

```
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat Berhenti berlangganan di Referensi AWS SDK for PHP API.

Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publikasikan pesan teks SMS

Contoh kode berikut menunjukkan cara mempublikasikan pesan SMS menggunakan Amazon SNS.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk detail API, lihat Publikasikan di Referensi AWS SDK for PHP API.

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SNS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Contoh Amazon SQS menggunakan SDK for PHP

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon SQS. AWS SDK for PHP

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SQS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
```

```
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk PHP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch SQS dengan Lambda menggunakan PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Migrasi dari Versi 2 AWS SDK for PHP

Topik ini menunjukkan cara memigrasikan kode Anda untuk menggunakan versi 3 AWS SDK for PHP dan bagaimana versi baru berbeda dari versi 2 SDK.

> **Note**
>
> Pola penggunaan dasar SDK (yaitu,`$result = $client->operation($params);`) tidak berubah dari versi 2 ke versi 3, yang akan menghasilkan migrasi yang lancar.

Pengantar

Versi 3 AWS SDK for PHP mewakili upaya signifikan untuk meningkatkan kemampuan SDK, menggabungkan lebih dari dua tahun umpan balik pelanggan, meningkatkan dependensi kami, meningkatkan kinerja, dan mengadopsi standar PHP terbaru.

Apa yang baru di versi 3?

Versi 3 AWS SDK for PHP mengikuti standar PSR-4 dan PSR-7 dan akan mengikuti standar ke SemVerdepan.

Fitur baru lainnya termasuk:

- Sistem middleware untuk menyesuaikan perilaku klien layanan
- Paginator fleksibel untuk iterasi melalui hasil paginasi
- Kemampuan untuk menanyakan data dari objek hasil dan paginator dengan JMESPath
- Debugging mudah melalui opsi `'debug'` konfigurasi

Lapisan HTTP terpisah

- Guzzle 6 digunakan secara default untuk mengirim permintaan, tetapi Guzzle 5 juga didukung.
- SDK akan bekerja di lingkungan di mana cURL tidak tersedia.
- Penangan HTTP kustom juga didukung.

Permintaan asinkron

- Fitur seperti pelayan dan pengunggah multibagian juga dapat digunakan secara asinkron.
- Alur kerja asinkron dapat dibuat menggunakan janji dan coroutine.
- Kinerja permintaan bersamaan atau batch ditingkatkan.

Apa yang berbeda dari versi 2?

Dependensi proyek diperbarui

Dependensi SDK telah berubah dalam versi ini.

- SDK sekarang membutuhkan PHP 8.1+. Kami menggunakan generator secara bebas dalam kode SDK.
- Kami telah memutakhirkan SDK untuk menggunakan Guzzle 6 (atau 5), yang menyediakan implementasi klien HTTP yang mendasari yang digunakan oleh SDK untuk mengirim permintaan ke layanan. AWS Versi terbaru Guzzle membawa serta sejumlah perbaikan, termasuk permintaan asinkron, penangan HTTP yang dapat ditukar, kepatuhan PSR-7, kinerja yang lebih baik, dan banyak lagi.
- Paket PSR-7 dari PHP-FIG (`psr/http-message`) mendefinisikan antarmuka untuk mewakili permintaan HTTP, respons HTTP, dan aliran. URLs Antarmuka ini digunakan di seluruh SDK dan Guzzle, yang menyediakan interoperabilitas dengan paket lain yang sesuai dengan PSR-7.
- Implementasi PSR-7 Guzzle (`guzzlehttp/psr7`) menyediakan implementasi antarmuka di PSR-7, dan beberapa kelas dan fungsi yang bermanfaat. Baik SDK dan Guzzle 6 sangat bergantung pada paket ini.
- Implementasi Promises/A+ Guzzle (`guzzlehttp/promises`) digunakan di seluruh SDK dan Guzzle untuk menyediakan antarmuka untuk mengelola permintaan dan coroutine asinkron. Sementara handler HTTP Multi-curl Guzzle pada akhirnya mengimplementasikan I/O model non-pemblokiran yang memungkinkan permintaan asinkron, paket ini menyediakan kemampuan untuk memprogram dalam paradigma itu. Lihat Janji di AWS SDK for PHP Versi 3 untuk lebih jelasnya.
- Implementasi PHP dari JMESPath(`mtdowling/jmespath.php`) digunakan dalam SDK untuk memberikan kemampuan query data dari `Aws\Result::search()` dan `Aws\ResultPaginator::search()` metode. Lihat JMESPath Ekspresi di AWS SDK for PHP Versi 3 untuk detail selengkapnya.

Opsi wilayah dan versi sekarang diperlukan

Saat membuat instance klien untuk layanan apa pun, tentukan `'region'` dan `'version'` opsi. Dalam versi 2 AWS SDK for PHP, `'version'` sepenuhnya opsional, dan `'region'` terkadang opsional. Di versi 3, keduanya selalu diperlukan. Menjadi eksplisit tentang kedua opsi ini memungkinkan Anda untuk mengunci ke dalam versi API dan AWS Wilayah yang Anda kodekan. Ketika versi API baru dibuat atau AWS Wilayah baru tersedia, Anda akan diisolasi dari perubahan yang berpotensi melanggar hingga Anda siap untuk memperbarui konfigurasi secara eksplisit.

> **Note**
>
> Jika Anda tidak khawatir tentang versi API yang Anda gunakan, Anda bisa mengatur `'version'` opsi ke`'latest'`. Namun, kami menyarankan Anda menetapkan nomor versi API secara eksplisit untuk kode produksi.
> Tidak semua layanan tersedia di semua AWS Wilayah. Anda dapat menemukan daftar Wilayah yang tersedia menggunakan referensi Wilayah dan Titik Akhir.
> Untuk layanan yang hanya tersedia melalui satu titik akhir global (misalnya, Amazon Route 53, dan Amazon CloudFront) AWS Identity and Access Management, buat instance klien dengan Wilayah yang dikonfigurasi disetel ke. `us-east-1`

> **Important**
>
> SDK juga mencakup klien multi-wilayah, yang dapat mengirimkan permintaan ke AWS Wilayah yang berbeda berdasarkan parameter (`@region`) yang disediakan sebagai parameter perintah. Wilayah yang digunakan secara default oleh klien ini ditentukan dengan `region` opsi yang diberikan ke konstruktor klien.

Instantiasi klien menggunakan konstruktor

Di versi 3 AWS SDK for PHP, cara Anda membuat instance klien telah berubah. Alih-alih `factory` metode di versi 2, Anda cukup membuat instance klien dengan menggunakan kata kunci. `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**
>
> Membuat instance klien menggunakan `factory()` metode ini masih berfungsi. Namun, itu dianggap usang.

Konfigurasi klien telah berubah

Opsi konfigurasi klien di versi 3 AWS SDK for PHP telah berubah sedikit dari versi 2. Lihat halaman Konfigurasi untuk AWS SDK for PHP Versi 3 untuk deskripsi semua opsi yang didukung.

> **Important**
>
> Di versi 3, `'key'` dan `'secret'` bukan lagi opsi yang valid di tingkat root, tetapi Anda dapat meneruskannya sebagai bagian dari `'credentials'` opsi. Salah satu alasan kami membuat ini adalah untuk mencegah pengembang dari hard-coding AWS kredensional mereka ke dalam proyek mereka.

Objek Sdk

Versi 3 AWS SDK for PHP memperkenalkan `Aws\Sdk` objek sebagai pengganti. `Aws\Common\Aws SdkObjek` bertindak sebagai pabrik klien dan digunakan untuk mengelola opsi konfigurasi bersama di beberapa klien.

Meskipun `Aws` kelas di versi 2 SDK bekerja seperti pencari lokasi layanan (selalu mengembalikan instance klien yang sama), `Sdk` kelas di versi 3 mengembalikan instance baru klien setiap kali digunakan.

`Sdk`Objek juga tidak mendukung format file konfigurasi yang sama dari SDK versi 2. Format konfigurasi itu khusus untuk Guzzle 3 dan sekarang sudah usang. Konfigurasi dapat dilakukan lebih sederhana dengan array dasar, dan didokumentasikan dalam Menggunakan Kelas Sdk.

Beberapa hasil API telah berubah

Untuk memberikan konsistensi dalam cara SDK mem-parsing hasil operasi API, Amazon, ElastiCache Amazon RDS, dan Amazon Redshift sekarang memiliki elemen pembungkus tambahan pada beberapa respons API.

Misalnya, memanggil DescribeEngineDefaultParametershasil Amazon RDS di versi 3 sekarang menyertakan elemen “EngineDefaults” pembungkus. Dalam versi 2, elemen ini tidak ada.

```
$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'];
```

Operasi berikut terpengaruh dan sekarang berisi elemen pembungkus dalam output hasil (disediakan di bawah dalam tanda kurung):

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
  - CopySnapshot (Cuplikan)
  - CreateCacheCluster (CacheCluster)
  - CreateCacheParameterGroup (CacheParameterGroup)
  - CreateCacheSecurityGroup (CacheSecurityGroup)
  - CreateCacheSubnetGroup (CacheSubnetGroup)
  - CreateReplicationGroup (ReplicationGroup)

- CreateSnapshot (Cuplikan)
- DeleteCacheCluster (CacheCluster)
- DeleteReplicationGroup (ReplicationGroup)
- DeleteSnapshot (Cuplikan)
- DescribeEngineDefaultParameters (EngineDefaults)
- ModifyCacheCluster (CacheCluster)
- ModifyCacheSubnetGroup (CacheSubnetGroup)
- ModifyReplicationGroup (ReplicationGroup)
- PurchaseReservedCacheNodesOffering (ReservedCacheNode)
- RebootCacheCluster (CacheCluster)
- RevokeCacheSecurityGroupIngress (CacheSecurityGroup)

- Amazon RDS
  - AddSourceIdentifierToSubscription (EventSubscription)
  - Otorisasi DBSecurity GroupIngress (DBSecurityGrup)
  - DBParameterGrup Salin (DBParameterGrup)
  - Salin DBSnapshot (DBSnapshot)
  - CopyOptionGroup (OptionGroup)
  - Buat DBInstance (DBInstance)
  - Buat DBInstance ReadReplica (DBInstance)
  - Buat DBParameter Grup (DBParameterGrup)
  - Buat DBSecurity Grup (DBSecurityGrup)
  - Buat DBSnapshot (DBSnapshot)
  - Buat DBSubnet Grup (DBSubnetGrup)
  - CreateEventSubscription (EventSubscription)
  - CreateOptionGroup (OptionGroup)
  - Hapus DBInstance (DBInstance)
  - Hapus DBSnapshot (DBSnapshot)
  - DeleteEventSubscription (EventSubscription)
  - DescribeEngineDefaultParameters (EngineDefaults)
  - Memodifikasi DBInstance (DBInstance)

- Ubah DBSubnet Grup (DBSubnetGrup)
- ModifyEventSubscription (EventSubscription)
- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesPenawaran (DicadangkanDBInstance)
- Reboot DBInstance (DBInstance)
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- Kembalikan DBInstance Dari DBSnapshot (DBInstance)
- Kembalikan DBInstance ToPointInTime (DBInstance)
- Cabut DBSecurity GroupIngress (Grup) DBSecurity

- Amazon Redshift
  - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
  - AuthorizeSnapshotAccess (Cuplikan)
  - CopyClusterSnapshot (Cuplikan)
  - CreateCluster (Kluster)
  - CreateClusterParameterGroup (ClusterParameterGroup)
  - CreateClusterSecurityGroup (ClusterSecurityGroup)
  - CreateClusterSnapshot (Cuplikan)
  - CreateClusterSubnetGroup (ClusterSubnetGroup)
  - CreateEventSubscription (EventSubscription)
  - CreateHsmClientCertificate (HsmClientCertificate)
  - CreateHsmConfiguration (HsmConfiguration)
  - DeleteCluster (Kluster)
  - DeleteClusterSnapshot (Cuplikan)
  - DescribeDefaultClusterParameters (DefaultClusterParameters)
  - DisableSnapshotCopy (Kluster)
  - EnableSnapshotCopy (Kluster)
  - ModifyCluster (Kluster)
  - ModifyClusterSubnetGroup (ClusterSubnetGroup)

- ModifyEventSubscription (EventSubscription)

- ModifySnapshotCopyRetentionPeriod (Kluster)
- PurchaseReservedNodeOffering (ReservedNode)
- RebootCluster (Kluster)
- RestoreFromClusterSnapshot (Kluster)
- RevokeClusterSecurityGroupIngress (ClusterSecurityGroup)
- RevokeSnapshotAccess (Cuplikan)
- RotateEncryptionKey (Kluster)

Kelas Enum telah dihapus

Kami telah menghapus `Enum` kelas (misalnya,`Aws\S3\Enum\CannedAcl`) yang ada di versi 2 AWS SDK for PHP. Enum adalah kelas konkret dalam API publik SDK yang berisi konstanta yang mewakili kelompok nilai parameter yang valid. Karena enum ini khusus untuk versi API, dapat berubah seiring waktu, dapat bertentangan dengan kata-kata yang dicadangkan PHP, dan akhirnya tidak terlalu berguna, kami telah menghapusnya di versi 3. Ini mendukung sifat agnostik versi berbasis data dan API versi 3.

Alih-alih menggunakan nilai dari `Enum` objek, gunakan nilai literal secara langsung (misalnya, `CannedAcl::PUBLIC_READ` →`'public-read'`).

Kelas pengecualian berbutir halus telah dihapus

Kami telah menghapus kelas pengecualian berbutir halus yang ada di ruang nama setiap layanan (misalnya,`Aws\Rds\Exception\{SpecificError}Exception`) untuk alasan yang sangat mirip ketika kami menghapus Enum. Pengecualian yang dilemparkan oleh layanan atau operasi bergantung pada versi API mana yang digunakan (mereka dapat berubah dari versi ke versi). Juga, daftar lengkap pengecualian yang dapat dilemparkan oleh operasi tertentu tidak tersedia, yang membuat kelas pengecualian berbutir halus versi 2 tidak lengkap.

Menangani kesalahan dengan menangkap kelas pengecualian root untuk setiap layanan (misalnya,`Aws\Rds\Exception\RdsException`). Anda dapat menggunakan `getAwsErrorCode()` metode pengecualian untuk memeriksa kode kesalahan tertentu. Ini secara fungsional setara dengan menangkap kelas pengecualian yang berbeda, tetapi menyediakan fungsi itu tanpa menambahkan bloat ke SDK.

Kelas fasad statis telah dihapus

Di versi 2 AWS SDK for PHP, ada fitur tidak jelas yang terinspirasi oleh Laravel yang memungkinkan Anda memanggil `enableFacades()` `Aws` kelas untuk mengaktifkan akses statis ke berbagai klien layanan. Fitur ini bertentangan dengan praktik terbaik PHP, dan kami berhenti mendokumentasikannya lebih dari setahun yang lalu. Di versi 3, fitur ini dihapus sepenuhnya. Ambil objek klien Anda dari `Aws\Sdk` objek dan gunakan sebagai instance objek, bukan kelas statis.

Paginator menggantikan iterator

Versi 2 AWS SDK for PHP memiliki fitur bernama * iterators*. Ini adalah objek yang digunakan untuk mengulangi hasil paginasi. Satu keluhan yang kami miliki tentang ini adalah bahwa mereka tidak cukup fleksibel, karena iterator hanya memancarkan nilai tertentu dari setiap hasil. Jika ada nilai lain yang Anda butuhkan dari hasil, Anda hanya dapat mengambilnya melalui pendengar acara.

Di versi 3, iterator telah diganti dengan Paginator. Tujuan mereka serupa, tetapi paginator lebih fleksibel. Ini karena mereka menghasilkan objek hasil alih-alih nilai dari respons.

Contoh berikut menunjukkan bagaimana paginator berbeda dari iterator, dengan mendemonstrasikan cara mengambil hasil paginasi untuk `S3 ListObjects` operasi di versi 2 dan versi 3.

```
// Version 2
$objects = $s3Client->getIterator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}
```

```
// Version 3
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results as $result) {
    // You can extract any data that you want from the result.
    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
}
```

Objek Paginator memiliki `search()` metode yang memungkinkan Anda untuk menggunakan JMESPathekspresi untuk mengekstrak data lebih mudah dari set hasil.

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
```

```
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

> **Note**
>
> `getIterator()`Metode ini masih didukung untuk memungkinkan transisi yang mulus ke versi 3, tetapi kami mendorong Anda untuk memigrasikan kode Anda untuk menggunakan paginator.

Banyak abstraksi tingkat tinggi telah berubah

Secara umum, banyak abstraksi tingkat yang lebih tinggi (objek pembantu khusus layanan, selain dari klien) telah ditingkatkan atau diperbarui. Beberapa bahkan telah dihapus.

- Diperbarui:
  - Cara Anda menggunakan Amazon S3 Multipart Upload telah berubah. Unggahan Multipart Amazon Glacier telah diubah dengan cara yang sama.
  - Cara membuat Amazon S3 pra-ditandatangani telah berubah URLs.
  - `Aws\S3\Sync`Namespace telah digantikan oleh kelas. `Aws\S3\Transfer` `S3Client::downloadBucket()`Metode `S3Client::uploadDirectory()` dan masih tersedia, tetapi memiliki opsi yang berbeda. Lihat dokumentasi untuk Amazon S3 Transfer Manager dengan AWS SDK for PHP Versi 3.
  - `Aws\S3\Model\ClearBucket`dan `Aws\S3\Model\DeleteObjectsBatch` telah digantikan oleh `Aws\S3\BatchDelete` dan`S3Client::deleteMatchingObjects()`.
  - Opsi dan perilaku untuk Menggunakan DynamoDB Session Handler AWS SDK for PHP dengan Versi 3 telah sedikit berubah.
  - `Aws\DynamoDb\Model\BatchRequest`Namespace telah digantikan oleh. `Aws\DynamoDb` `\WriteRequestBatch` Lihat dokumentasi untuk DynamoDB WriteRequestBatch.
  - `Aws\Ses\SesClient`Sekarang menangani pengkodean base64 `RawMessage` saat menggunakan `SendRawEmail` operasi.
- Dihapus:
  - Amazon `Item` DynamoDB`Attribute`,, `ItemIterator` dan kelas - Ini sebelumnya tidak digunakan lagi di Versi 2.7.0.

- Validator pesan Amazon SNS - Ini sekarang merupakan proyek terpisah dan ringan yang tidak memerlukan SDK sebagai dependensi. Proyek ini, bagaimanapun, termasuk dalam distribusi Phar dan ZIP SDK. Anda dapat menemukan panduan memulai di blog Pengembangan AWS PHP.
- Amazon S3 `AcpBuilder` dan objek terkait telah dihapus.

Membandingkan sampel kode dari kedua versi SDK

Contoh berikut menunjukkan beberapa cara di mana menggunakan versi 3 AWS SDK for PHP berbeda dari versi 2.

Contoh: Operasi Amazon S3 **`ListObjects`**

Dari versi 2 SDK

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = S3Client::factory([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

Dari versi 3 SDK

Perbedaan utama:

- Gunakan `new` alih-alih `factory()` untuk membuat instance klien.
- `'region'`Opsi `'version'` dan diperlukan selama instantiasi.

```
<?php

require '/path/to/vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$s3 = new S3Client([
    'profile' => 'my-credential-profile',
    'region'  => 'us-east-1',
    'version' => '2006-03-01'
]);

try {
    $result = $s3->listObjects([
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'    => 'my-object-key'
    ]);

    foreach ($result['Contents'] as $object) {
        echo $object['Key'] . "\n";
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . "\n";
}
```

Contoh: Membuat instantiasi klien dengan konfigurasi global

Dari versi 2 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.
```

Dari versi 3 SDK

Perbedaan utama:

- Gunakan `Aws\Sdk` kelas sebagai ganti`Aws\Common\Aws`.
- Tidak ada file konfigurasi. Gunakan array untuk konfigurasi sebagai gantinya.
- `'version'`Opsi ini diperlukan selama instantiasi.
- Gunakan `create<Service>()` metode alih-alih`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.
```

Keamanan untuk AWS SDK for PHP

Keamanan cloud di Amazon Web Services (AWS) merupakan prioritas tertinggi. Sebagai seorang pelanggan AWS , Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan dari organisasi yang paling sensitif terhadap keamanan. Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model Tanggung Jawab Bersama menggambarkan ini sebagai Keamanan dari Cloud dan Keamanan dalam Cloud.

Security of the Cloud - AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan semua layanan yang ditawarkan di AWS Cloud dan memberi Anda layanan yang dapat Anda gunakan dengan aman. Tanggung jawab keamanan kami adalah prioritas tertinggi di AWS, dan efektivitas keamanan kami secara teratur diuji dan diverifikasi oleh auditor pihak ketiga sebagai bagian dari Program AWS Kepatuhan.

Keamanan di Cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan, dan faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

Topik

- Perlindungan data dalam AWS SDK for PHP Versi 3
- Identity and Access Management
- Validasi Kepatuhan untuk AWS Produk atau Layanan ini
- Ketahanan untuk AWS Produk atau Layanan ini
- Keamanan Infrastruktur untuk AWS Produk atau Layanan ini
- Migrasi Klien Enkripsi Amazon S3 (V1 ke V2) di Versi 3 AWS SDK for PHP
- Migrasi Klien Enkripsi Amazon S3 (V2 ke V3) di Versi 3 AWS SDK for PHP

Perlindungan data dalam AWS SDK for PHP Versi 3

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang

perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS SDK for PHP Versi 3 atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Identity and Access Management

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang

dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Layanan AWS bekerja dengan IAM
- Memecahkan masalah AWS identitas dan akses

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan. AWS

Pengguna layanan — Jika Anda menggunakan Layanan AWS untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur AWS, lihat Memecahkan masalah AWS identitas dan akses atau panduan pengguna yang Layanan AWS Anda gunakan.

Administrator layanan — Jika Anda bertanggung jawab atas AWS sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS. Tugas Anda adalah menentukan AWS fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AWS, lihat panduan pengguna yang Layanan AWS Anda gunakan.

Administrator IAM – Jika Anda adalah administrator IAM, Anda sebaiknya mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke AWS. Untuk melihat contoh kebijakan AWS berbasis identitas yang dapat Anda gunakan di IAM, lihat panduan pengguna yang Anda gunakan. Layanan AWS

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensil dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensil. Google/ Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensil dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk

informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna gabungan, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon. EC2 Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari

cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat Ringkasan daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .

- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Layanan AWS bekerja dengan IAM

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Untuk mempelajari cara menggunakan spesifik Layanan AWS dengan IAM, lihat bagian keamanan dari Panduan Pengguna layanan yang relevan.

Memecahkan masalah AWS identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di AWS
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di AWS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `awes:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `awes:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran AWS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang

dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah AWS mendukung fitur-fitur ini, lihatBagaimana Layanan AWS bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Validasi Kepatuhan untuk AWS Produk atau Layanan ini

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Ketahanan untuk AWS Produk atau Layanan ini

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones.

Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan.

Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Keamanan Infrastruktur untuk AWS Produk atau Layanan ini

AWS Produk atau layanan ini menggunakan layanan terkelola, dan karenanya dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Produk atau Layanan ini melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan principal IAM. Atau Anda dapat menggunakan AWS Security Token

Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

AWS Produk atau layanan ini mengikuti model tanggung jawab bersama melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat halaman dokumentasi keamanan AWS layanan dan AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan.

Migrasi Klien Enkripsi Amazon S3 (V1 ke V2) di Versi 3 AWS SDK for PHP

> **Note**
>
> Jika Anda menggunakan Versi 2 (V2) dari klien enkripsi Amazon S3 dan ingin bermigrasi ke Versi 3 (V3), lihat. Migrasi Klien Enkripsi Amazon S3 (V2 ke V3) di Versi 3 AWS SDK for PHP

Topik ini menunjukkan cara memigrasikan aplikasi Anda dari klien enkripsi Amazon Simple Storage Service (Amazon S3) ke Versi 1 (V1) Amazon Simple Storage Service (Amazon S3) ke Versi 2 (V2), dan memastikan ketersediaan aplikasi selama proses migrasi.

Ikhtisar migrasi

Migrasi ini terjadi dalam dua fase:

1. Perbarui klien yang ada untuk membaca format baru. Pertama, gunakan versi terbaru dari aplikasi AWS SDK for PHP Anda. Ini memungkinkan klien enkripsi V1 yang ada untuk mendekripsi objek yang ditulis oleh klien V2 baru. Jika aplikasi Anda menggunakan beberapa AWS SDKs, Anda harus memutakhirkan setiap SDK secara terpisah.

2. Migrasikan enkripsi dan dekripsi klien ke V2. Setelah semua klien enkripsi V1 Anda dapat membaca format baru, Anda dapat memigrasikan klien enkripsi dan dekripsi yang ada ke versi V2 masing-masing.

Perbarui klien yang ada untuk membaca format baru

Klien enkripsi V2 menggunakan algoritma enkripsi yang tidak didukung oleh versi klien yang lebih lama. Langkah pertama dalam migrasi adalah memperbarui klien dekripsi V1 Anda ke rilis SDK

terbaru. Setelah menyelesaikan langkah ini, klien V1 aplikasi Anda akan dapat mendekripsi objek yang dienkripsi oleh klien enkripsi V2. Lihat detail di bawah untuk setiap versi utama dari AWS SDK for PHP.

Upgrade AWS SDK for PHP Versi 3

Versi 3 adalah versi terbaru dari AWS SDK for PHP. Untuk menyelesaikan migrasi ini, Anda harus menggunakan versi 3.148.0 atau yang lebih baru dari paket. `aws/aws-sdk-php`

Instalasi dari Command Line

Untuk proyek yang diinstal menggunakan Composer, dalam file Composer, perbarui paket SDK ke versi 3.148.0 SDK dan kemudian jalankan perintah berikut.

```
composer update aws/aws-sdk-php
```

Menginstal Menggunakan File Phar atau Zip

Gunakan salah satu metode berikut. Pastikan untuk menempatkan file SDK yang diperbarui di lokasi yang diperlukan oleh kode Anda, yang ditentukan oleh pernyataan require.

Untuk proyek yang diinstal menggunakan file Phar, unduh file yang diperbarui: `aws.phar`.

```
<?php
  require '/path/to/aws.phar';
?>
```

Untuk proyek yang diinstal menggunakan file Zip, unduh file yang diperbarui: .

```
<?php
  require '/path/to/aws-autoloader.php';
?>
```

Migrasikan enkripsi dan dekripsi klien ke V2

Setelah memperbarui klien Anda untuk membaca format enkripsi baru, Anda dapat memperbarui aplikasi Anda ke klien enkripsi dan dekripsi V2. Langkah-langkah berikut menunjukkan cara berhasil memigrasikan kode Anda dari V1 ke V2.

Persyaratan untuk Memperbarui ke Klien V2

1. Konteks AWS KMS enkripsi harus diteruskan ke dalam `S3EncryptionClientV2::putObject` dan `S3EncryptionClientV2::putObjectAsync` metode. AWS KMS konteks enkripsi adalah array asosiatif pasangan kunci-nilai, yang harus Anda tambahkan ke konteks enkripsi untuk enkripsi kunci. AWS KMS Jika tidak ada konteks tambahan yang diperlukan, Anda dapat meneruskan array kosong.

2. `@SecurityProfile`harus diteruskan ke dalam `getObject` dan `getObjectAsync` metode di`S3EncryptionClientV2`. `@SecurityProfile`adalah parameter wajib baru dari `getObject...` metode ini. Jika disetel ke`'V2'`, hanya objek yang dienkripsi dalam format yang kompatibel dengan V2 yang dapat didekripsi. Menyetel parameter ini `'V2_AND_LEGACY'` juga memungkinkan objek yang dienkripsi dalam format yang kompatibel dengan V1 untuk didekripsi. Untuk mendukung migrasi, setel `@SecurityProfile` ke`'V2_AND_LEGACY'`. Gunakan `'V2'` hanya untuk pengembangan aplikasi baru.

3. (opsional) Sertakan `@KmsAllowDecryptWithAnyCmk` parameter dalam `S3EncryptionClientV2::getObject` dan `S3EncryptionClientV2::getObjectAsync*` `methods.` Parameter baru telah ditambahkan dipanggil`@KmsAllowDecryptWithAnyCmk`. Mengatur parameter ini untuk `true` mengaktifkan dekripsi tanpa memasok kunci KMS. Nilai default-nya adalah `false`.

4. Untuk dekripsi dengan klien V2, jika `@KmsAllowDecryptWithAnyCmk` parameter tidak disetel ke `true` untuk panggilan `"getObject..."` metode, a `kms-key-id` harus diberikan ke konstruktor. `KmsMaterialsProviderV2`

Contoh migrasi

Contoh 1: Migrasi ke klien V2

Pra-migrasi

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
```

```
    ])
);
```

Pasca-migrasi

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Contoh 2: Menggunakan AWS KMS dengan kms-key-id

> **Note**
>
> Contoh-contoh ini menggunakan impor dan variabel yang didefinisikan dalam Contoh 1. Misalnya, `$encryptionClient`.

Pra-migrasi

```
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,
]);
```

Pasca-migrasi

```
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,
]);
```

Migrasi Klien Enkripsi Amazon S3 (V2 ke V3) di Versi 3 AWS SDK for PHP

> **Note**
>
> Jika Anda menggunakan Versi 1 (V1) klien enkripsi Amazon S3, Anda harus terlebih dahulu bermigrasi ke Versi 2 (V2) sebelum bermigrasi ke Versi 3 (V3). Lihat Migrasi Klien Enkripsi Amazon S3 (V1 ke V2) di Versi 3 AWS SDK for PHP.

Topik ini menunjukkan cara memigrasikan aplikasi Anda dari klien enkripsi Amazon Simple Storage Service (Amazon S3) ke Versi 2 (V3) Amazon Simple Storage Service (Amazon S3) ke Versi 3 (V3), dan memastikan ketersediaan aplikasi selama proses migrasi. Versi 3 memperkenalkan AES GCM dengan Kebijakan Komitmen dan Komitmen Utama untuk meningkatkan keamanan dan melindungi terhadap gangguan kunci data.

Ikhtisar migrasi

Migrasi ini terjadi dalam dua fase:

1. Perbarui klien yang ada untuk membaca format baru. Pertama, gunakan versi terbaru dari aplikasi AWS SDK for PHP Anda. Ini memungkinkan klien enkripsi V2 yang ada untuk mendekripsi objek

yang ditulis oleh klien V3 baru. Jika aplikasi Anda menggunakan beberapa AWS SDKs, Anda harus memutakhirkan setiap SDK secara terpisah.

2. Migrasikan enkripsi dan dekripsi klien ke V3. Setelah semua klien enkripsi V2 Anda dapat membaca format baru, Anda dapat memigrasikan klien enkripsi dan dekripsi yang ada ke versi V3 masing-masing.

Memahami konsep V3

Versi 3 dari klien enkripsi Amazon S3 memperkenalkan dua peningkatan keamanan utama: Kebijakan Komitmen dan AES GCM dengan algoritma Komitmen Kunci. Memahami konsep-konsep ini sangat penting untuk migrasi yang sukses.

Kebijakan Komitmen

Kebijakan Komitmen mengontrol cara klien enkripsi menangani komitmen utama selama operasi enkripsi dan dekripsi. Versi 3 menyediakan tiga opsi kebijakan:

`FORBID_ENCRYPT_ALLOW_DECRYPT`

Perilaku enkripsi: Mengenkripsi objek tanpa komitmen utama.

Perilaku dekripsi: Memungkinkan dekripsi objek yang dienkripsi dengan atau tanpa komitmen kunci.

Implikasi keamanan: Kebijakan ini tidak memberlakukan komitmen utama pada objek yang baru dienkripsi, yang memungkinkan perusakan kunci data. Gunakan kebijakan ini hanya selama fase migrasi awal ketika Anda perlu mempertahankan kompatibilitas dengan klien V2.

Kompatibilitas versi: Objek yang dienkripsi dengan kebijakan ini dapat dibaca oleh semua implementasi V2 dan V3.

`REQUIRE_ENCRYPT_ALLOW_DECRYPT`

Perilaku enkripsi: Mengenkripsi objek dengan komitmen utama menggunakan algoritma. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

Perilaku dekripsi: Memungkinkan dekripsi objek yang dienkripsi dengan atau tanpa komitmen kunci.

Implikasi keamanan: Kebijakan ini memberikan keamanan yang ditingkatkan untuk objek yang baru dienkripsi sambil mempertahankan kemampuan untuk membaca objek yang ada. Ini adalah kebijakan yang direkomendasikan untuk sebagian besar skenario migrasi.

Kompatibilitas versi: Objek yang dienkripsi dengan kebijakan ini hanya dapat dibaca oleh V3 dan implementasi V2 terbaru.

Pertimbangan migrasi: Sebelum menggunakan kebijakan ini, pastikan semua klien yang perlu membaca objek terenkripsi telah ditingkatkan ke V3 atau V2 terbaru.

`REQUIRE_ENCRYPT_REQUIRE_DECRYPT`

Perilaku enkripsi: Mengenkripsi objek dengan komitmen utama menggunakan algoritma. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

Perilaku dekripsi: Hanya memungkinkan dekripsi objek yang dienkripsi dengan komitmen utama. Objek yang dienkripsi tanpa komitmen kunci akan gagal didekripsi.

Implikasi keamanan: Kebijakan ini memberikan tingkat keamanan tertinggi dengan menegakkan komitmen utama untuk enkripsi dan dekripsi. Gunakan kebijakan ini hanya setelah semua objek dimigrasi untuk menggunakan komitmen utama.

Kompatibilitas versi: Hanya implementasi V3 yang dapat menggunakan kebijakan ini. Mencoba mendekripsi objek terenkripsi V1 atau V2 dengan kebijakan ini akan gagal.

Pertimbangan migrasi: Kebijakan ini hanya boleh digunakan setelah menyelesaikan migrasi penuh dan mengenkripsi ulang semua objek yang ada dengan komitmen utama.

AES GCM dengan Komitmen Utama

Algoritma AES GCM with Key Commitment (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) adalah algoritma enkripsi baru yang diperkenalkan di V3 yang memberikan perlindungan terhadap serangan gangguan kunci data.

Peningkatan keamanan: `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` melindungi terhadap gangguan kunci data dengan mengikat kunci data secara kriptografis ke konten terenkripsi. Ini mencegah penyerang mengganti kunci data yang berbeda selama dekripsi, yang dapat menyebabkan dekripsi data yang tidak diinginkan.

Kompatibilitas versi: Objek yang dienkripsi hanya `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` dapat didekripsi oleh V3 dan implementasi V2 terbaru dari klien enkripsi Amazon S3. Klien V1 tidak dapat mendekripsi objek yang dienkripsi dengan algoritma ini.

> **Important**
>
> Persyaratan peningkatan: Sebelum mengaktifkan enkripsi dengan `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (dengan menggunakan kebijakan REQUIRE_ENCRYPT_ALLOW_DECRYPT atau REQUIRE_ENCRYPT_REQUIRE_DECRYPT), Anda harus memastikan bahwa semua klien yang perlu membaca objek terenkripsi telah ditingkatkan ke V3. Kegagalan untuk meng-upgrade semua pembaca akan mengakibatkan kegagalan dekripsi untuk objek yang dienkripsi dengan komitmen utama.

Perbarui klien yang ada untuk membaca format baru

Klien enkripsi V3 menggunakan algoritma enkripsi dan fitur komitmen utama yang tidak didukung oleh versi klien yang lebih lama. Langkah pertama dalam migrasi adalah memperbarui klien dekripsi V2 Anda ke rilis SDK terbaru. Setelah menyelesaikan langkah ini, klien V2 aplikasi Anda akan dapat mendekripsi objek yang dienkripsi oleh klien enkripsi V3. Lihat detail di bawah untuk setiap metode instalasi AWS SDK for PHP.

Membangun dan menginstal versi SDK terbaru

Untuk menyelesaikan migrasi ini, Anda harus menggunakan versi terbaru dari `aws/aws-sdk-php` paket yang menyertakan dukungan klien enkripsi V3.

Instalasi dari Komposer

Untuk proyek yang diinstal menggunakan Composer, dalam file Composer, perbarui paket SDK ke versi terbaru SDK dan kemudian jalankan perintah berikut.

```
composer update aws/aws-sdk-php
```

Menginstal Menggunakan File Phar atau Zip

Gunakan salah satu metode berikut. Pastikan untuk menempatkan file SDK yang diperbarui di lokasi yang diperlukan oleh kode Anda, yang ditentukan oleh pernyataan require.

Untuk proyek yang diinstal menggunakan file Phar, unduh file yang diperbarui: `aws.phar`.

```
<?php
```

```
  require '/path/to/aws.phar';
?>
```

Untuk proyek yang diinstal menggunakan file Zip, unduh file yang diperbarui: .

```
<?php
  require '/path/to/aws-autoloader.php';
?>
```

Membangun, menginstal, dan menyebarkan aplikasi

Setelah memperbarui SDK, buat kembali dan terapkan ulang aplikasi Anda untuk memastikan semua komponen menggunakan versi yang diperbarui. Langkah ini sangat penting untuk memastikan bahwa klien V2 Anda dapat membaca objek yang dienkripsi oleh klien V3.

Ikuti prosedur penerapan standar organisasi Anda untuk meluncurkan aplikasi yang diperbarui. Pastikan semua instance aplikasi Anda diperbarui sebelum melanjutkan untuk memigrasikan klien enkripsi dan dekripsi Anda ke V3.

Setelah penerapan, verifikasi bahwa aplikasi Anda masih dapat mendekripsi objek yang ada dan tidak ada kesalahan yang terjadi selama operasi normal. Ini mengonfirmasi bahwa pembaruan SDK berhasil dan aplikasi Anda siap untuk fase migrasi berikutnya.

Migrasikan klien Enkripsi dan Dekripsi ke V3

Setelah memperbarui klien Anda untuk membaca format enkripsi baru, Anda dapat memperbarui aplikasi Anda ke klien enkripsi dan dekripsi V3. Contoh berikut menunjukkan cara memigrasi kode Anda dari V2 ke V3 dengan sukses.

Menggunakan Klien Enkripsi V3

V3 memperkenalkan `S3EncryptionClientV3` kelas dan `KmsMaterialsProviderV3` untuk menggantikan setara V2. Perbedaan utama dalam V3 adalah:

- V3 menggunakan `KmsMaterialsProviderV3` (sama seperti V2) tetapi memverifikasi konteks enkripsi saat mendekripsi objek dalam panggilan. `GetObject`
- V3 memperkenalkan Kebijakan Komitmen untuk mengontrol perilaku enkripsi dan dekripsi.

Contoh: Migrasi dari V2 ke V3 dengan Enkripsi KMS

Pra-migrasi (V2)

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$encryptionClient = new S3EncryptionClientV2(
   new S3Client([
      'profile' => 'default',
      'region' => 'us-east-1',
      'version' => 'latest',
   ])
);

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
   new KmsClient([
      'profile' => 'default',
      'region' => 'us-east-1',
      'version' => 'latest',
   ]),
   $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
   'Cipher' => 'gcm',
   'KeySize' => 256,
];

$encryptionClient->putObject([
   '@MaterialsProvider' => $materialsProvider,
   '@CipherOptions' => $cipherOptions,
   '@KmsEncryptionContext' => ['context-key' => 'context-value'],
   'Bucket' => $bucket,
   'Key' => $key,
   'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
   '@KmsAllowDecryptWithAnyCmk' => true,
   '@SecurityProfile' => 'V2_AND_LEGACY',
```

```
    '@CommitmentPolicy' => 'FORBID_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Selama migrasi (V3 dengan kompatibilitas mundur)

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
```

```
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Pasca-migrasi (V3 dengan komitmen utama)

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);
```

```
$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
   'Cipher' => 'gcm',
   'KeySize' => 256,
];

$encryptionClient->putObject([
   '@MaterialsProvider' => $materialsProvider,
   '@CipherOptions' => $cipherOptions,
   // Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
   // This encrypts with key commitment and does not decrypt V2 objects
   '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
   '@KmsEncryptionContext' => ['context-key' => 'context-value'],
   'Bucket' => $bucket,
   'Key' => $key,
   'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
   '@SecurityProfile' => 'V3',
   // Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
   // This encrypts with key commitment and does not decrypt V2 objects
   '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
   '@MaterialsProvider' => $materialsProvider,
   '@CipherOptions' => $cipherOptions,
   'Bucket' => $bucket,
   'Key' => $key,
]);
```

Perbedaan utama dalam V3:

- Gunakan `KmsMaterialsProviderV3` alih-alih `KmsMaterialsProviderV2`
- `@KmsEncryptionContext`Parameter masih diperlukan untuk `putObject` operasi
- `@KmsEncryptionContext`Parameter ini opsional untuk `getObject` operasi dan akan memverifikasi bahwa konteks enkripsi yang disediakan cocok dengan yang ada di objek.
- `@SecurityProfile`Parameter mengontrol versi enkripsi mana yang dapat didekripsi. Setel `'V3_AND_LEGACY'` untuk mendukung membaca objek terenkripsi V1 dan V2 selama migrasi

- `@CommitmentPolicy`Parameter mengontrol kebijakan komitmen untuk operasi ini. Setel `'FORBID_ENCRYPT_ALLOW_DECRYPT'` untuk mendukung membaca objek terenkripsi non komitmen selama migrasi

Contoh tambahan

Contoh berikut menunjukkan opsi konfigurasi tambahan yang tersedia di V3 yang dapat membantu Anda mengelola proses migrasi dan mengontrol perilaku enkripsi.

Mengaktifkan dukungan lama

Selama migrasi, Anda mungkin perlu mendekripsi objek yang dienkripsi dengan V1 atau V2 dari klien enkripsi Amazon S3. `@SecurityProfile`Parameter mengontrol versi enkripsi mana yang dapat didekripsi klien V3 Anda.

Kapan menggunakan konfigurasi ini: Gunakan profil `'V3_AND_LEGACY'` keamanan saat aplikasi Anda perlu membaca objek yang dienkripsi oleh klien V1 atau V2. Ini biasa terjadi selama periode migrasi ketika Anda memiliki campuran objek terenkripsi lama dan baru di ember Anda.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

```
$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

// Decrypt objects encrypted with V1, V2, or V3
$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

`@SecurityProfile`Parameter menerima nilai-nilai berikut:

- `'V3'`(default): Hanya mendekripsi objek yang dienkripsi dengan V3 menggunakan komitmen utama
- `'V3_AND_LEGACY'`: Dekripsi objek yang dienkripsi dengan V1, V2, atau V3

> **Important**
>
> Setelah menyelesaikan migrasi dan mengenkripsi ulang semua objek dengan V3, Anda harus menghapus `@SecurityProfile` parameter atau mengaturnya `'V3'` untuk memastikan keamanan maksimum.

Mengkonfigurasi metode penyimpanan

Klien enkripsi Amazon S3 dapat menyimpan metadata enkripsi dengan dua cara: di header metadata objek atau dalam file instruksi terpisah. `@MetadataStrategy`Parameter mengontrol metode penyimpanan mana yang digunakan.

Kapan menggunakan konfigurasi ini: Gunakan `'INSTRUCTION_FILE'` saat Anda perlu mempertahankan metadata objek asli atau saat bekerja dengan objek yang memiliki batasan ukuran

metadata. Gunakan `'METADATA'` (default) untuk penerapan yang lebih sederhana di mana metadata enkripsi dapat disimpan di samping objek.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
    'KeySize' => 256,
];

// Store encryption metadata in a separate instruction file
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'INSTRUCTION_FILE',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
```

```
]);

// Store encryption metadata in object headers (default)
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'METADATA',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

`@MetadataStrategy`Parameter menerima nilai-nilai berikut:

- `'METADATA'`(default): Simpan metadata enkripsi di header metadata objek
- `'INSTRUCTION_FILE'`: Simpan metadata enkripsi dalam file instruksi terpisah dengan akhiran `.instruction`

> **Note**
>
> Saat menggunakan`'INSTRUCTION_FILE'`, algoritma AES GCM dengan Key Commitment memberikan perlindungan tambahan terhadap gangguan kunci data. Objek yang menggunakan `'METADATA'` penyimpanan tidak mendapat manfaat dari perlindungan tambahan ini.

FAQ untuk AWS SDK for PHP Versi 3

Metode apa yang tersedia pada klien?

AWS SDK for PHP Menggunakan deskripsi layanan dan metode magic __call () dinamis untuk menjalankan operasi API. Anda dapat menemukan daftar lengkap metode yang tersedia untuk klien layanan web dalam dokumentasi API klien.

Apa yang harus saya lakukan tentang kesalahan sertifikat cURL SSL?

Masalah ini dapat terjadi saat menggunakan bundel out-of-date CA dengan cURL dan SSL. Anda dapat mengatasi masalah ini dengan memperbarui bundel CA di server Anda atau mengunduh bundel up-to-date CA lainnya dari situs web cURL secara langsung.

Secara default, AWS SDK for PHP akan menggunakan bundel CA yang dikonfigurasi ketika PHP dikompilasi. Anda dapat mengubah bundel CA default yang digunakan oleh PHP dengan memodifikasi pengaturan konfigurasi `openssl.cafile` PHP .ini untuk diatur ke jalur file CA pada disk.

Versi API apa yang tersedia untuk klien?

Diperlukan `version` opsi saat membuat klien. Daftar versi API yang tersedia dapat ditemukan di setiap halaman dokumentasi API klien::aws-php-class:<index.html>. Jika Anda tidak dapat memuat versi API tertentu, Anda mungkin perlu memperbarui salinan AWS SDK for PHP.

Anda dapat memberikan string `latest` ke nilai konfigurasi “versi” untuk menggunakan versi API terbaru yang tersedia yang dapat ditemukan oleh penyedia API klien Anda (api_provider default akan memindai `src/data` direktori SDK untuk model API).

> **Warning**
>
> Kami tidak menyarankan penggunaan `latest` dalam aplikasi produksi karena menarik versi minor SDK baru yang menyertakan pembaruan API dapat merusak aplikasi produksi Anda.

Versi Region apa yang tersedia untuk klien?

`region`Opsi diperlukan saat membuat klien, dan ditentukan menggunakan nilai string. Untuk daftar AWS Wilayah dan titik akhir yang tersedia, lihat AWS Wilayah dan Titik Akhir di. Referensi Umum AWS

```
// Set the Region to the EU (Frankfurt) Region.
$s3 = new Aws\S3\S3Client([
    'region'  => 'eu-central-1',
    'version' => '2006-03-01'
]);
```

Mengapa saya tidak dapat mengunggah atau mengunduh file yang lebih besar dari 2 GB?

Karena tipe integer PHP ditandatangani, dan banyak platform menggunakan bilangan bulat 32-bit, AWS SDK for PHP tidak benar menangani file yang lebih besar dari 2 GB pada tumpukan 32-bit (di mana “tumpukan” termasuk CPU, OS, server web, dan biner PHP). Ini adalah masalah PHP yang terkenal. Dalam kasus Microsoft Windows, hanya build PHP 7 yang mendukung bilangan bulat 64-bit.

Solusi yang disarankan adalah menggunakan tumpukan Linux 64-bit, seperti AMI Linux Amazon 64-bit, dengan versi PHP terbaru diinstal.

Untuk informasi selengkapnya, lihat PHP filesize: Mengembalikan nilai.

Bagaimana saya bisa melihat data apa yang dikirim melalui kabel?

Anda bisa mendapatkan informasi debug, termasuk data yang dikirim melalui kawat, menggunakan `debug` opsi di konstruktor klien. Ketika opsi ini diatur `ketrue`, semua mutasi perintah yang sedang dijalankan, permintaan dikirim, respons yang diterima, dan hasil yang sedang diproses dipancarkan ke STDOUT. Ini termasuk data yang dikirim dan diterima melalui kabel.

```
$s3Client = new Aws\S3\S3Client([
    'region'  => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

Bagaimana saya bisa mengatur header arbitrer berdasarkan permintaan?

Anda dapat menambahkan header arbitrer apa pun ke operasi layanan dengan menambahkan middleware khusus ke atau. `Aws\HandlerList` `Aws\CommandInterface` `Aws\ClientInterface` Contoh berikut menunjukkan cara menambahkan `X-Foo-Baz` header ke `PutObject` operasi Amazon S3 tertentu menggunakan metode `Aws\Middleware::mapRequest` helper.

Lihat MapRequest untuk informasi selengkapnya.

Bagaimana saya bisa menandatangani permintaan sewenang-wenang?

Anda dapat menandatangani permintaan arbitrer:aws-php-class: PSR-7 <Class-PSR.http.Message.RequestInterface.html> menggunakan kelas SDK:aws-php-class: SignatureV4. <class-Aws.Signature.SignatureV4.html>

Lihat Menandatangani Permintaan CloudSearch Domain Amazon Kustom dengan AWS SDK for PHP Versi 3 untuk contoh lengkap tentang cara melakukannya.

Bagaimana saya bisa memodifikasi perintah sebelum mengirimnya?

Anda dapat memodifikasi perintah sebelum mengirimnya dengan menambahkan middleware kustom ke `Aws\HandlerList` atau`Aws\CommandInterface`. `Aws\ClientInterface` Contoh berikut menunjukkan cara menambahkan parameter perintah khusus ke perintah sebelum dikirim, pada dasarnya menambahkan opsi default. Contoh ini menggunakan metode `Aws\Middleware::mapCommand` helper.

Lihat MapCommand untuk informasi selengkapnya.

Apa itu CredentialsException?

Jika Anda melihat `Aws\Exception\CredentialsException` saat menggunakan AWS SDK for PHP, itu berarti SDK tidak diberikan kredensyal apa pun dan tidak dapat menemukan kredensyal di lingkungan.

Jika Anda membuat instance klien tanpa kredensyal, saat pertama kali Anda melakukan operasi layanan, SDK akan mencoba menemukan kredensyal. Pertama memeriksa di beberapa variabel lingkungan tertentu, kemudian mencari kredensyal profil misalnya, yang hanya tersedia pada instance Amazon EC2 yang dikonfigurasi. Jika sama sekali tidak ada kredensil yang diberikan atau ditemukan, an `Aws\Exception\CredentialsException` dilemparkan.

Jika Anda melihat kesalahan ini dan Anda bermaksud menggunakan kredensyal profil instans, Anda harus memastikan bahwa EC2 instans Amazon yang menjalankan SDK dikonfigurasi dengan peran IAM yang sesuai.

Jika Anda melihat kesalahan ini dan Anda tidak bermaksud menggunakan kredensyal profil instans, Anda harus memastikan bahwa Anda memberikan kredensyal dengan benar ke SDK.

Untuk informasi selengkapnya, lihat Kredensial untuk AWS SDK for PHP Versi 3.

Apakah AWS SDK for PHP pekerjaan pada HHVM?

Saat ini AWS SDK for PHP tidak berjalan di HHVM, dan tidak akan dapat sampai masalah dengan semantik hasil di HHVM diselesaikan.

Bagaimana cara menonaktifkan SSL?

Anda dapat menonaktifkan SSL dengan mengatur `scheme` parameter dalam metode pabrik klien ke 'http'. Penting untuk dicatat bahwa tidak semua layanan mendukung `http` akses. Lihat AWS Wilayah dan Titik Akhir dalam Referensi Umum AWS daftar wilayah, titik akhir, dan skema yang didukung.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

Warning

Karena SSL memerlukan semua data untuk dienkripsi dan membutuhkan lebih banyak paket TCP untuk menyelesaikan jabat tangan koneksi daripada hanya TCP, menonaktifkan SSL dapat memberikan peningkatan kinerja yang kecil. Namun, dengan SSL dinonaktifkan, semua data dikirim melalui kabel yang tidak dienkripsi. Sebelum menonaktifkan SSL, Anda

harus mempertimbangkan dengan cermat implikasi keamanan dan potensi penyadapan melalui jaringan.

Apa yang harus saya lakukan tentang “Parse error”?

Mesin PHP akan melempar kesalahan parsing ketika menemukan sintaks yang tidak dimengerti. Ini hampir selalu ditemui ketika mencoba menjalankan kode yang ditulis untuk versi PHP yang berbeda.

Jika Anda mengalami kesalahan penguraian, periksa sistem Anda dan pastikan sistem tersebut memenuhi Persyaratan dan Rekomendasi SDK untuk Versi 3. AWS SDK for PHP

Mengapa klien Amazon S3 mendekompresi file gzip?

Beberapa penangan HTTP, termasuk handler HTTP Guzzle 6 default, akan mengembang badan respons terkompresi secara default. Anda dapat mengganti perilaku ini dengan menyetel opsi HTTP decode_content ke. `false` Untuk alasan kompatibilitas mundur, default ini tidak dapat diubah, tetapi kami menyarankan Anda menonaktifkan decoding konten di tingkat klien S3.

Lihat decode_content untuk contoh cara menonaktifkan decoding konten otomatis.

Bagaimana cara menonaktifkan penandatanganan tubuh di Amazon S3?

Anda dapat menonaktifkan penandatanganan tubuh dengan mengatur `ContentSHA256` parameter di objek perintah ke`Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`. Kemudian AWS SDK for PHP akan menggunakannya sebagai header 'x-amz-content-sha-256' dan checksum tubuh dalam permintaan kanonik.

```
$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);
```

Bagaimana skema coba lagi ditangani di? AWS SDK for PHP

AWS SDK for PHP Memiliki a `RetryMiddleware` yang menangani perilaku coba lagi. Dalam hal kode status HTTP 5xx untuk kesalahan server, SDK mencoba ulang pada 500, 502, 503 dan 504.

Pengecualian pelambatan, termasuk,`RequestLimitExceeded`,,`Throttling`, `RequestThrottled` dan `ProvisionedThroughputExceededException` `ThrottlingExceptionBandwidthLimitExceeded`, juga ditangani dengan percobaan ulang.

Ini AWS SDK for PHP juga mengintegrasikan penundaan eksponensial dengan algoritma backoff dan jitter dalam skema coba lagi. Selanjutnya, perilaku coba ulang default dikonfigurasi `3` untuk semua layanan kecuali Amazon DynamoDB, yaitu. `10`

Bagaimana cara menangani pengecualian dengan kode kesalahan?

Selain `Exception` kelas AWS SDK for PHP-customized, setiap klien AWS layanan memiliki kelas pengecualian sendiri yang mewarisi dari. AwsExceptionAwsException Anda dapat menentukan jenis kesalahan yang lebih spesifik untuk ditangkap dengan kesalahan khusus API yang tercantum di bawah `Errors` bagian setiap metode.

Informasi kode kesalahan tersedia dengan getAwsErrorCode () from`Aws\Exception` `\AwsException`.

```
$sns = new \Aws\Sns\SnsClient([
    'region' => 'us-west-2',
    'version' => 'latest',
]);

try {
```

```
    $sns->publish([
        // parameters
        ...
    ]);
    // Do something
} catch (SnsException $e) {
    switch ($e->getAwsErrorCode()) {
        case 'EndpointDisabled':
        case 'NotFound':
            // Do something
            break;
    }
}
```

Glosarium

Versi API

Layanan memiliki satu atau beberapa versi API, dan versi mana yang Anda gunakan menentukan operasi dan parameter mana yang valid. Versi API diformat seperti tanggal. Misalnya, versi API terbaru untuk Amazon S3 adalah. `2006-03-01` Tentukan versi saat Anda mengonfigurasi objek klien.

Klien

Objek klien digunakan untuk menjalankan operasi untuk layanan. Setiap layanan yang didukung dalam SDK memiliki objek klien yang sesuai. Objek klien memiliki metode yang sesuai one-to-one dengan operasi layanan. Lihat the section called “Membuat klien layanan dasar” untuk mulai menggunakan klien.

Perintah

Objek perintah merangkum eksekusi operasi. Ketika Anda menjalankan operasi layanan seperti yang dijelaskan di bagian permintaan pembuatan, Anda tidak akan berurusan langsung dengan objek perintah. Objek perintah dapat diakses menggunakan `getCommand()` metode klien, untuk menggunakan fitur-fitur canggih SDK seperti permintaan bersamaan dan batching. Lihat Command Objects dalam panduan AWS SDK for PHP Versi 3 untuk lebih jelasnya.

Handler

Handler adalah fungsi yang melakukan transformasi aktual dari perintah dan permintaan menjadi hasil. Handler biasanya mengirimkan permintaan HTTP. Handler dapat disusun dengan middleware untuk meningkatkan perilaku mereka. Handler adalah fungsi yang menerima `Aws\CommandInterface` dan a `Psr\Http\Message\RequestInterface` dan mengembalikan janji yang dipenuhi dengan `Aws\ResultInterface` atau ditolak dengan alasan. `Aws\Exception\AwsException`

JMESPath

JMESPathadalah bahasa kueri untuk data seperti JSON. AWS SDK for PHP Menggunakan JMESPath ekspresi untuk query struktur data PHP. JMESPath ekspresi dapat digunakan langsung pada `Aws\Result` dan `Aws\ResultPaginator` objek melalui `search($expression)` metode.

Middleware

Middleware adalah jenis khusus dari fungsi tingkat tinggi yang menambah perilaku mentransfer perintah dan mendelegasikan ke handler “berikutnya”. Fungsi middleware menerima `Aws\CommandInterface` dan a `Psr\Http\Message\RequestInterface` dan mengembalikan janji yang dipenuhi dengan `Aws\ResultInterface` atau ditolak dengan alasan. `Aws\Exception\AwsException`

Operasi

Mengacu pada operasi tunggal dalam API layanan (misalnya, `CreateTable` untuk DynamoDB, `RunInstances` untuk Amazon). EC2 Dalam SDK, operasi dijalankan dengan memanggil metode dengan nama yang sama pada objek klien layanan yang sesuai. Menjalankan operasi melibatkan mempersiapkan dan mengirim permintaan HTTP ke layanan dan mengurai respons. Proses mengeksekusi operasi ini diabstraksikan oleh SDK melalui objek perintah.

Paginator

Beberapa operasi AWS layanan diberi paginasi dan merespons dengan hasil terpotong. Misalnya, `ListObjects` operasi Amazon S3 hanya mengembalikan hingga 1000 objek sekaligus. Operasi seperti ini memerlukan pembuatan permintaan berikutnya dengan parameter token (atau penanda) untuk mengambil seluruh rangkaian hasil. paginator adalah fitur SDK yang bertindak sebagai abstraksi atas proses ini untuk memudahkan pengembang menggunakan paginated. APIs Mereka diakses melalui `getPaginator()` metode klien. Lihat Paginator di panduan AWS SDK for PHP Versi 3 untuk detail selengkapnya.

Janji

Janji merupakan hasil akhir dari operasi asinkron. Cara utama berinteraksi dengan janji adalah melalui metode saat itu, yang mendaftarkan panggilan balik untuk menerima nilai akhir janji atau alasan mengapa janji tidak dapat dipenuhi.

Wilayah

Layanan didukung di satu atau lebih wilayah geografis. Layanan mungkin berbeda endpoints/ URLs di setiap wilayah, yang ada untuk mengurangi latensi data dalam aplikasi Anda. Berikan wilayah saat Anda mengonfigurasi objek klien, sehingga SDK dapat menentukan titik akhir mana yang akan digunakan dengan layanan.

SDK

Istilah “SDK” dapat merujuk ke AWS SDK for PHP perpustakaan secara keseluruhan, tetapi juga mengacu pada **`Aws\Sdk`** kelas (dokumen), yang bertindak sebagai pabrik untuk objek klien untuk

setiap layanan. `Sdk`Kelas ini juga memungkinkan Anda menyediakan satu set nilai konfigurasi global yang diterapkan ke semua objek klien yang dibuatnya.

Layanan

Cara umum untuk merujuk ke salah satu AWS layanan (misalnya, Amazon S3, Amazon DynamoDB, dll.) AWS OpsWorks. Setiap layanan memiliki objek klien yang sesuai di SDK yang mendukung satu atau beberapa versi API. Setiap layanan juga memiliki satu atau lebih operasi yang membentuk API-nya. Layanan didukung di satu atau beberapa wilayah.

Tanda tangan

Saat menjalankan operasi, SDK menggunakan kredensil Anda untuk membuat tanda tangan digital permintaan Anda. Layanan kemudian memverifikasi tanda tangan sebelum memproses permintaan Anda. Proses penandatanganan dienkapsulasi oleh SDK, dan terjadi secara otomatis menggunakan kredensil yang Anda konfigurasikan untuk klien.

Pelayan

Pelayan adalah fitur SDK yang membuatnya lebih mudah untuk bekerja dengan operasi yang mengubah status sumber daya dan yang pada akhirnya konsisten atau tidak sinkron. Misalnya, operasi Amazon `CreateTable` DynamoDB mengirimkan respons kembali segera, tetapi tabel mungkin tidak siap untuk diakses selama beberapa detik. Mengeksekusi pelayan memungkinkan Anda untuk menunggu sampai sumber daya masuk ke keadaan tertentu dengan tidur dan memilih status sumber daya. Pelayan diakses menggunakan `waitUntil()` metode klien. Lihat panduan Pelayan di AWS SDK for PHP Versi 3 untuk lebih jelasnya.

Untuk AWS terminologi terbaru, lihat AWS Glosarium di. Referensi Umum AWS

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting sejak rilis terakhir Panduan AWS SDK for PHP Pengembang.

Perubahan terbaru:

Perubahan	Deskripsi	Tanggal
Daftar isi pembaruan	Daftar isi direvisi agar lebih konsisten dengan panduan SDK lainnya.	Juli 28, 2025
Perlindungan integritas data dengan checksum	Konten diperbarui dengan rincian tentang perhitungan checksum otomatis.	Januari 16, 2025
Revisi topik kredensil	Topik direorganisasi. Detail selengkapnya disediakan untuk rantai penyedia kredensi default.	Januari 10, 2025
Migrasi V3 klien enkripsi Amazon S3	Menambahkan topik tentang migrasi klien enkripsi Amazon S3 dari V2 ke V3	Desember 4, 2024
Unggahan multipart Amazon S3	Dokumentasikan array 'params' yang dapat digunakan untuk mengkonfi gurasi subperintah dari dan `ObjectUploader` `MultipartUploader`	6 November 2024
Pengunggah objek	Memperjelas penggunaan callback yang tersedia pada upload `ObjectUploader` untuk S3	Oktober 11, 2024

Perbarui nama bucket Amazon S3	Nama bucket S3 yang diperbarui di seluruh panduan.	September 30, 2024
Titik akhir EventBridge global Amazon	Tambahkan contoh kode yang menunjukkan cara menggunakan titik akhir EventBridge global Amazon	22 Desember 2023
AWS Runtime Umum (AWS CRT)	Tambahkan topik yang membahas penggunaan AWS Common Runtime (AWS CRT) oleh SDK for PHP.	17 November 2023
StreamWrapper mkdir () pembaruan	Tambahkan informasi tentang bekerja dengan ember dan objek folder dengan menggunakan`mkdir()`.	November 2, 2023
Pembuatan klien layanan	Perbarui cuplikan kode dengan menghapus parameter 'versi' karena 'terbaru' adalah default.	31 Agustus 2023
Pembuatan klien dasar	Perbarui cuplikan kode dengan menghapus parameter 'versi' karena 'terbaru' adalah default.	31 Agustus 2023
Daftar isi	Daftar isi yang diperbarui untuk membuat contoh kode lebih mudah diakses.	1 Juni 2023
Pembaruan praktik terbaik IAM	Memperbarui panduan untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi lebih lanjut, lihat Praktik terbaik keamanan di IAM. Pembaruan untuk Memulai.	Mei 20, 2023

Unggahan multipart Amazon S3	Termasuk informasi konfigurasi untuk unggahan sinkron. Menambahkan opsi `add_content_md5` unggah untuk unggahan asinkron.	13 April 2023
Transfer direktori Amazon S3	Menambahkan opsi `add_content_md5` transfer.	13 April 2023
Informasi referensi	Menambahkan beberapa tautan ke konten detail yang relevan di Panduan Referensi Alat AWS SDKs dan Alat. Pemformatan panduan yang diperbarui.	14 September 2022
Pembersihan umum	Menambahkan referensi ke AWS SDKs and Tools Reference Guide. AWS Key Management Service Bagian yang diperbarui untuk mencerminkan pembaruan terminologi.	23 Agustus 2022
Bekerja dengan AWS layanan	Termasuk daftar contoh kode yang tersedia di GitHub.	1 April 2022
Mengaktifkan metrik SDK	Menghapus informasi tentang mengaktifkan metrik SDK, yang tidak digunakan lagi pada 20 Desember 2021.	27 Januari 2022
Migrasi klien enkripsi Amazon S3	Menambahkan topik tentang migrasi klien enkripsi Amazon S3	7 Agustus 2020

Perubahan lama:

Ubah	Deskripsi	Tanggal rilis
Contoh Secrets Manager	Tambahkan lebih banyak contoh layanan	27 Maret 2019
Penemuan titik akhir	Konfigurasi penemuan titik akhir	15 Februari 2019
Amazon CloudFront	Tambahkan lebih banyak contoh layanan	25 Jan 2019
Fitur layanan	Metrik SDK	11 Jan 2018
Amazon Kinesis, Amazon SNS	Tambahkan lebih banyak contoh layanan	14 Desember 2018
Contoh Amazon SES	Tambahkan lebih banyak contoh layanan	5 Okt 2018
AWS KMS contoh	Tambahkan lebih banyak contoh layanan	8 Agustus 2018
Kredensial	Mengklarifikasi dan menyederhanakan panduan kredensional	30 Juni 2018
MediaConvert contoh	Tambahkan lebih banyak contoh layanan	15 Juni 2018
Tata letak web baru	Dokumentasi beralih ke gaya AWS	9 Mei 2018
Enkripsi Amazon S3	Enkripsi sisi klien	17 November 2017
Amazon S3, Amazon SQS	Tambahkan lebih banyak contoh layanan	26 Mar 2017
Amazon S3, IAM, Amazon EC2	Tambahkan lebih banyak contoh layanan	17 Mar 2017

Ubah	Deskripsi	Tanggal rilis
Tambahkan kredensil	Menambahkan dukungan untuk AssumeRole dan ini	17 Jan 2017
Contoh S3	S3 Multi-Region dan posting yang telah ditentukan sebelumnya	18 Mar 2016
OpenSearch Layanan dan Amazon CloudSearch	Tambahkan lebih banyak contoh layanan	28 Desember 2015
Baris perintah	Tambahkan parameter perintah	13 Agustus 2015
Fitur layanan	Menambahkan fitur layanan untuk S3 dan AWS	30 Apr 2015
Versi SDK baru	Versi 3 dari yang AWS SDK for PHP dirilis.	26 Mei 2015

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.