Guida per gli sviluppatori

AWS SDK per PHP

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK per PHP: Guida per gli sviluppatori

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Che cos'è la AWS SDK per PHP? 1
Inizia a usare l'SDK 1
Risorse aggiuntive 1
Documentazione API 2
Manutenzione e supporto per le versioni principali dell'SDK 2
Inizia a usare 3
Autenticazione SDK con AWS 3
Avvia una sessione del portale di AWS accesso 4
Scopri di più sull'autenticazione 5
Prerequisiti 5
Requisiti 6
Raccomandazioni 6
Test di compatibilità 7
Installazione dell'SDK 7
Installa come dipendenza tramite Composer AWS SDK per PHP 8
Installazione utilizzando il pacchetto phar 9
Installazione utilizzando il file ZIP 9
Ciao tutorial 10
Inclusione dell'SDK nel codice 10
Scrivi il codice 10
Esecuzione del programma 11
Passaggi successivi 11
Da utilizzare AWS Cloud9 con l'SDK 12
Passaggio 1: configura il tuo Account AWS da utilizzare AWS Cloud9 12
Fase 2: configura il tuo ambiente di AWS Cloud9 sviluppo 13
Fase 3: Configurare il AWS SDK per PHP 13
Passaggio 4: Scarica il codice di esempio 14
Passaggio 5: Esegui codice di esempio 15
Configurare l'SDK 17
Utilizzo di base 17
Prerequisiti 17
Inclusione dell'SDK nel codice 10
Riepilogo dell'utilizzo 18
Creazione di un client 18

Utilizzo della Sdk classe ... 19
Esecuzione delle operazioni di servizio ... 20
Richieste asincrone ... 22
Lavorare con oggetti di risultato ... 24
Gestione degli errori ... 25
Opzioni di configurazione ... 27
api_provider ... 29
credenziali ... 29
debug ... 31
statistiche ... 33
endpoint ... 35
endpoint_provider ... 35
endpoint_discovery ... 36
handler ... 37
http ... 38
http_handler ... 46
profilo ... 48
Regione ... 48
retries ... 49
scheme ... 52
service ... 52
signature_provider ... 53
signature_version ... 53
ua_append ... 54
use_aws_shared_config_files ... 54
validate ... 54
version ... 55
Credenziali ... 56
Precedenza delle impostazioni ... 56
Collabora con i fornitori di credenziali ... 57
Assumi un ruolo IAM ... 72
Usa credenziali temporanee da AWS STS ... 79
Crea clienti anonimi ... 81
Oggetti di comando ... 82
Uso implicito dei comandi ... 82
Parametri di comando ... 83

Creazione di oggetti di comando ... 83
Comando `HandlerList` ... 84
`CommandPool` ... 85
Promesse ... 89
Cos'è una promessa? ... 89
Promesse in SDK ... 90
Concatenare le promesse ... 92
Aspettando le promesse ... 93
Annullamento delle promesse ... 94
Combinare promesse ... 94
Gestori e middleware ... 97
Gestori ... 97
Middleware ... 99
Creazione di gestori personalizzati ... 106
Streams ... 107
Decoratori Stream ... 108
Impaginatori ... 112
Oggetti Paginator ... 113
Enumerazione dei dati dai risultati ... 113
Impaginazione asincrona ... 114
Waiter ... 115
Configurazione del cameriere ... 116
Attesa in modo asincrono ... 117
JMESPath espressioni ... 119
Estrazione di dati dai risultati ... 119
Estrazione di dati dagli impaginatori ... 124
Utilizzate l'estensione AWS CRT ... 124
Ho bisogno dell'estensione AWS CRT? ... 124
Come si installa l'estensione AWS CRT? ... 125
Aggiornamento dalla versione 2 ... 125
Introduzione ... 125
Novità della versione 3 ... 125
Cosa cambia rispetto alla versione 2 ... 126
Confronto tra gli esempi di codice da parte di entrambe le versioni dell'SDK ... 135
Condivisi `config` e file `credentials` ... 138
Profili denominati ... 139

Lavora con AWS i servizi 140
Usa le funzionalità e le opzioni 140
Amazon DynamoDB 140
Amazon S3 148
Esempi di codice con indicazioni 171
Credenziali 171
CloudFront Esempi di Amazon 172
Amazon CloudSearch 201
CloudWatch Esempi di Amazon 203
EC2 Esempi di Amazon 227
OpenSearch Servizio Amazon 241
AWS Identity and Access Management esempi 242
AWS Key Management Service 267
Esempi di Kinesis 289
AWS Elemental MediaConvert 305
Esempi di Amazon S3 312
AWS Secrets Manager 347
Esempi di Amazon SES 356
Esempi di Amazon SNS 388
Esempi di Amazon SQS 407
Amazon EventBridge 420
Esempi di codice 422
API Gateway 423
Azioni 423
Scenari 429
Aurora 429
Scenari 429
Auto Scaling 430
Nozioni di base 432
Azioni 423
Amazon Bedrock 446
Azioni 423
Runtime di Amazon Bedrock 448
Scenari 429
Amazon Nova 450
Amazon Titan Image Generator 452

Anthropic Claude 453
Diffusione stabile 455
Amazon DocumentDB 456
Esempi serverless 456
DynamoDB 458
Nozioni di base 432
Azioni 423
Scenari 429
Esempi serverless 456
Amazon EC2 490
Azioni 423
AWS Glue 495
Nozioni di base 432
Azioni 423
IAM 516
Nozioni di base 432
Azioni 423
Kinesis 532
Esempi serverless 456
AWS KMS 536
Nozioni di base 432
Azioni 423
Lambda 572
Nozioni di base 432
Azioni 423
Scenari 429
Esempi serverless 456
MSK Amazon 602
Esempi serverless 456
Amazon RDS 604
Azioni 423
Scenari 429
Esempi serverless 456
Servizi di dati di Amazon RDS 612
Scenari 429
Amazon Rekognition 614

Scenari ... 429
Amazon S3 ... 615
Nozioni di base ... 432
Azioni ... 423
Scenari ... 429
Esempi serverless ... 456
Bucket di directory S3 ... 638
Nozioni di base ... 432
Amazon SES ... 654
Scenari ... 429
Amazon SNS ... 655
Azioni ... 423
Scenari ... 429
Esempi serverless ... 456
Amazon SQS ... 675
Esempi serverless ... 456
Sicurezza ... 680
Protezione dei dati ... 680
Identity and Access Management ... 681
Destinatari ... 682
Autenticazione con identità ... 682
Gestione dell'accesso con policy ... 686
Come Servizi AWS lavorare con IAM ... 689
Risoluzione dei problemi di AWS identità e accesso ... 689
Convalida della conformità ... 691
Resilienza ... 692
Sicurezza dell'infrastruttura ... 693
Migrazione del client di crittografia Amazon S3 ... 693
Panoramica della migrazione ... 693
Aggiorna i client esistenti per leggere nuovi formati ... 694
Migra i client di crittografia e decrittografia alla V2 ... 695
Esempi di migrazione ... 695
Domande frequenti ... 699
Quali metodi sono disponibili su un client? ... 699
Cosa devo fare in caso di errore cURL di un certificato SSL? ... 699
Quali versioni dell'API sono disponibili per un client? ... 699

Quali versioni della regione sono disponibili per un client? 700
Perché non riesco a caricare o scaricare i file di dimensioni superiori a 2 GB? 700
Come è possibile controllare i dati che vengono trasmessi in rete? 700
Come è possibile impostare delle intestazioni arbitrarie su una richiesta? 701
Come posso firmare una richiesta arbitraria? 701
Come posso modificare un comando prima di inviarlo? 701
Che cos'è un CredentialsException? 701
Funziona su HHVM? AWS SDK per PHP 702
Come posso disabilitare il protocollo SSL? 702
Cosa devo fare se visualizzo un "Errore di analisi"? 703
Perché il client Amazon S3 decomprime i file gzip? 703
Come faccio a disabilitare la firma del corpo in Amazon S3? 703
Come viene gestito lo schema dei nuovi tentativi nell' AWS SDK per PHP? 704
Come posso gestire le eccezioni con codici di errore? 704
Glossario 706
Cronologia dei documenti 709
..... dccxiv

Cos'è la AWS SDK per PHP versione 3?

La AWS SDK per PHP versione 3 consente agli sviluppatori PHP di utilizzare Amazon Web Services nel loro codice PHP e di creare applicazioni e software affidabili utilizzando servizi come Amazon S3, Amazon DynamoDB e S3 Glacier. Puoi iniziare in pochi minuti installando l'SDK tramite Composer, richiedendo il pacchetto, oppure scaricando la versione standalone o il `aws/aws-sdk-php` file. `aws.zip` `aws.phar`

Non tutti i servizi sono immediatamente disponibili nell'SDK. Per scoprire quali servizi sono attualmente supportati da AWS SDK per PHP, consulta Nome del servizio e versione dell'API.

> **Note**
>
> Se stai migrando il codice dalla versione 2 dell'SDK alla versione 3, assicurati di leggere Upgrade from Version 2 di. AWS SDK per PHP

Inizia a usare l'SDK

Se sei pronto a provare l'SDK, segui il capitolo. Inizia a usare Ti guida attraverso l'autenticazione AWS, la configurazione del tuo ambiente di sviluppo e la creazione della tua prima applicazione di base con Amazon S3.

Risorse aggiuntive

- DOMANDE FREQUENTI
- Glossario
- AWS SDKs e Guida di riferimento agli strumenti: contiene impostazioni, funzionalità e altri concetti fondamentali comuni a. AWS SDKs
- Documentazione Guzzle
- Esempi di codice che utilizzano sono disponibili nel repository awsdocs/ aws-doc-sdk-examples. AWS SDK per PHP
- La community PHP SDK su Gitter.
- AWS re:Post.

GitHub:

- Il codice sorgente di AWS SDK per PHP è disponibile nel repository aws/. aws-sdk-php
- Contribuire all'SDK
- Segnala un bug o richiedi una funzionalità

Documentazione API

Trova la documentazione API per l'SDK su latest/reference/. https://docs.aws.amazon.com/sdk-for-php/

Manutenzione e supporto per le versioni principali dell'SDK

Per informazioni sulla manutenzione e il supporto per le versioni principali dell'SDK e le relative dipendenze sottostanti, consulta quanto segue nella and Tools Reference Guide:AWS SDKs

- AWS SDKs e politica di manutenzione degli strumenti
- AWS SDKs e la matrice di supporto delle versioni di Tools

Inizia a usare

Questo capitolo è dedicato a consentirti di iniziare a utilizzare la AWS SDK per PHP versione 3.

Argomenti

- Autenticazione SDK con AWS
- Requisiti e raccomandazioni per la AWS SDK per PHP versione 3
- Installa la AWS SDK per PHP versione 3
- Hello tutorial per AWS SDK per PHP
- Utilizzare AWS Cloud9 con AWS SDK per PHP

Autenticazione SDK con AWS

È necessario stabilire in che modo il codice si autentica durante lo sviluppo con AWS . Servizi AWSÈ possibile configurare l'accesso programmatico alle AWS risorse in diversi modi a seconda dell'ambiente e dell' AWS accesso a disposizione.

Per scegliere il metodo di autenticazione e configurarlo per l'SDK, consulta Autenticazione e accesso nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

Consigliamo ai nuovi utenti che si stanno sviluppando localmente e che non dispongono di un metodo di autenticazione da parte del datore di lavoro di AWS IAM Identity Center configurarlo. Questo metodo include l'installazione di AWS CLI per facilitare la configurazione e per accedere regolarmente al portale di AWS accesso. Se scegli questo metodo, l'ambiente dovrebbe contenere i seguenti elementi dopo aver completato la procedura per l'autenticazione di IAM Identity Center nella AWS SDKs and Tools Reference Guide:

- Il AWS CLI, che viene utilizzato per avviare una sessione del portale di AWS accesso prima di eseguire l'applicazione.
- Un AWS`config`file condiviso con un `[default]` profilo con un set di valori di configurazione a cui può fare riferimento l'SDK. Per trovare la posizione di questo file, consulta Posizione dei file condivisi nella Guida di riferimento agli strumenti AWS SDKs e strumenti.
- Il `config` file condiviso contiene l'`region`impostazione. Questo imposta l'impostazione predefinita Regione AWS utilizzata dall'SDK per le richieste. Questa regione viene utilizzata per le richieste di servizio SDK che non sono configurate in modo esplicito con una proprietà. `region`

- L'SDK utilizza la configurazione del provider di token SSO del profilo per acquisire le credenziali prima di inviare richieste a. AWS Il `sso_role_name` valore, che è un ruolo IAM connesso a un set di autorizzazioni IAM Identity Center, consente l'accesso ai dati Servizi AWS utilizzati nell'applicazione.

  Il seguente `config` file di esempio mostra un profilo predefinito impostato con la configurazione del provider di token SSO. L'`sso_session`impostazione del profilo si riferisce alla `sso-session`sezione denominata. La `sso-session` sezione contiene le impostazioni per avviare una sessione del portale di AWS accesso.

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

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

AWS SDK per PHP Non è necessario aggiungere pacchetti aggiuntivi (come `SSO` e`SSOOIDC`) all'applicazione per utilizzare l'autenticazione IAM Identity Center.

Avvia una sessione del portale di AWS accesso

Prima di eseguire un'applicazione che consente l'accesso Servizi AWS, è necessaria una sessione attiva del portale di AWS accesso affinché l'SDK utilizzi l'autenticazione IAM Identity Center per risolvere le credenziali. A seconda della durata della sessione configurata, l'accesso alla fine scadrà e l'SDK riscontrerà un errore di autenticazione. Per accedere al portale di AWS accesso, esegui il seguente comando in. AWS CLI

```
aws sso login
```

Se hai seguito le istruzioni e disponi di una configurazione predefinita del profilo, non è necessario richiamare il comando con un'`--profile`opzione. Se la configurazione del provider di token SSO utilizza un profilo denominato, il comando è `aws sso login --profile named-profile`.

Per verificare facoltativamente se hai già una sessione attiva, esegui il AWS CLI comando seguente.

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

Se la sessione è attiva, la risposta a questo comando riporta l'account IAM Identity Center e il set di autorizzazioni configurati nel `config` file condiviso.

> **Note**
>
> Se hai già una sessione attiva del portale di AWS accesso ed eseguia`ws sso login`, non ti verrà richiesto di fornire credenziali.
> La procedura di accesso potrebbe richiedere all'utente di consentire l' AWS CLI accesso ai dati. Poiché AWS CLI è basato sull'SDK per Python, i messaggi di autorizzazione potrebbero contenere variazioni del `botocore` nome.

Scopri di più sull'autenticazione

- Per maggiori dettagli sull'utilizzo di IAM Identity Center per l'autenticazione, consulta Understand IAM Identity Center Authentication nella AWS SDKs and Tools Reference Guide
- Per ulteriori informazioni sulle best practice, consulta Best practice per la sicurezza in IAM nella Guida per l'utente di IAM.
- Per creare AWS credenziali a breve termine, consulta Temporary Security Credentials nella IAM User Guide.
- Per ulteriori informazioni su altri fornitori di credenziali che AWS SDK per PHP possono utilizzare, consulta Fornitori di credenziali standardizzati nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti.

Requisiti e raccomandazioni per la AWS SDK per PHP versione 3

Per ottenere risultati ottimali AWS SDK per PHP, assicuratevi che il vostro ambiente supporti i seguenti requisiti e consigli.

Requisiti

Per utilizzare il AWS SDK per PHP, è necessario utilizzare la versione PHP 8.1 o successiva con l'estensione PHP SimpleXML abilitata. Se devi firmare un account Amazon privato CloudFront URLs, ti serve anche l'estensione PHP OpenSSL.

Raccomandazioni

Oltre ai requisiti minimi, si consiglia inoltre di installare, disinstallare e utilizzare quanto segue.

Installare cURL 7.16.2 o la versione successiva	Utilizzare una versione recente di cURL compilata con OpenSSL/NSS e zlib. Se cURL non è installato sul sistema e non si intende configurare un gestore http personalizzato per il client, l'SDK utilizza il wrapper di flusso PHP.
Utilizzare OPCache	Usa l' OPcache estensione per migliorar e le prestazioni di PHP memorizzando il bytecode degli script precompilati nella memoria condivisa. Ciò evita che PHP debba caricare e analizzare gli script in ogni richiesta . Questa estensione è solitamente abilitata per impostazione predefinita. Quando esegui Amazon Linux, devi installare il pacchetto yum php56-opcache o php55-opc ache per utilizzare l'estensione. OPCache
Disinstalla Xdebug negli ambienti di produzione	Xdebug può aiutare a identificare i colli di bottiglia delle prestazioni. Tuttavia, se le prestazioni sono fondamentali per la propria applicazione, non installare l'estensione Xdebug nell'ambiente di produzione. Caricare l'estensione rallenta notevolmente le prestazio ni dell'SDK.
Usare un autoloader classmap Composer	Gli autoloader caricano le classi come richiesto da uno script PHP. Composer genera un

	autoloader che possa caricare gli script PHP dell'applicazione e tutti gli altri script PHP necessari per l'applicazione, tra cui l' AWS SDK per PHP. Per gli ambienti di produzione, è consiglia bile utilizzare un autoloader classmap per migliorare le prestazioni. È possibile generare un autoloader classmap trasferendo l'opzione `-o` o `==optimize-autoloader` al comando di installazione del Composer.

Test di compatibilità

Esegui il `compatibility-test.php`file che si trova nella base di codice SDK per verificare che il tuo sistema sia in grado di eseguire l'SDK. Oltre a soddisfare i requisiti minimi di sistema dell'SDK, il test di compatibilità controlla le impostazioni facoltative e fornisce suggerimenti che possono aiutare a migliorare le prestazioni. Il test di compatibilità fornisce risultati per la riga di comando o per un browser web. Durante la revisione dei risultati del test in un browser, i controlli con esito positivo sono visualizzati in verde, le avvertenze in viola e gli errori in rosso. Quando si effettua l'esecuzione dalla riga di comando, il risultato di un controllo viene visualizzato su una riga separata.

Quando si segnala un problema con l'SDK, condividere l'output del test di compatibilità consente di identificare la causa sottostante.

Installa la AWS SDK per PHP versione 3

Puoi installare la AWS SDK per PHP versione 3:

- Come dipendenza tramite Composer
- Come un file phar contenuto nell'SDK
- Come un file ZIP del kit SDK

Prima di installare la AWS SDK per PHP versione 3, assicurati che il tuo ambiente utilizzi la versione PHP 5.5 o successiva. Scopri di più sui requisiti e sui consigli relativi all'ambiente.

> **Note**
>
> L'installazione dell'SDK tramite i metodi .phar e.zip richiede l'installazione e l'attivazione dell'estensione PHP Multibyte String separatamente.

Installa come dipendenza tramite Composer AWS SDK per PHP

Composer è il metodo consigliato per installare. AWS SDK per PHP Composer è uno strumento per PHP in grado di gestire e installare le dipendenze del progetto.

Per ulteriori informazioni su come installare Composer, configurare il caricamento automatico e seguire le altre best practice per la definizione delle dipendenze, consulta getcomposer.org.

Installa Composer

Se Composer non è già presente nel progetto, scarica e installa Composer dalla pagina Scarica Composer.

- Per Windows, seguite le istruzioni di Windows Installer.
- Per Linux, segui le istruzioni di installazione della riga di comando.

Aggiungi AWS SDK per PHP come dipendenza tramite Composer

Se Composer è già installato globalmente sul sistema, esegui quanto segue nella directory di base del progetto per installarlo AWS SDK per PHP come dipendenza:

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

Altrimenti, digitate questo comando Composer per installare la versione più recente di AWS SDK per PHP come dipendenza.

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

Aggiungi l'autoloader ai tuoi script php

L'installazione di Composer crea diverse cartelle e file nell'ambiente. Il file principale che utilizzerai è `autoload.php`, che si trova nella cartella `vendor` dell'ambiente.

Per utilizzarli AWS SDK per PHP nei tuoi script, includi il caricatore automatico negli script, come segue.

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

Installazione utilizzando il pacchetto phar

Ogni versione di AWS SDK per PHP include un phar (archivio PHP) preconfezionato che contiene tutte le classi e le dipendenze necessarie per eseguire l'SDK. Inoltre, il phar registra automaticamente un caricatore automatico di classi per tutte le sue dipendenze. AWS SDK per PHP

Puoi scaricare il file phar contenuto nel pacchetto e includerlo negli script.

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

> **Note**
>
> L'utilizzo di PHP con la patch Suhosin non è consigliato, ma è comune nelle distribuzioni Ubuntu e Debian. In questo caso, potresti dover abilitare l'uso degli archivi phar nel file suhosin.ini. Se non effettui questa operazione, l'inclusione di un file phar nel codice causerà un errore che non verrà segnalato. Per modificare il file suhosin.ini, aggiungi la riga seguente.
>
> ```
> suhosin.executor.include.whitelist = phar
> ```

Installazione utilizzando il file ZIP

AWS SDK per PHP Include un file ZIP contenente tutte le classi e le dipendenze necessarie per eseguire l'SDK. Inoltre, il file ZIP include un autoloader della classe per l' AWS SDK per PHP e tutte le relative dipendenze.

Per installare l'SDK, scarica il file .zip ed estrailo nel progetto in un percorso di tua scelta. Quindi, includi l'autoloader negli script, come descritto di seguito.

```
<?php
```

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

Hello tutorial per AWS SDK per PHP

Dai il benvenuto ad Amazon S3 utilizzando il. AWS SDK per PHP L'esempio seguente mostra un elenco dei tuoi bucket Amazon S3.

Inclusione dell'SDK nel codice

Qualsiasi tecnica abbia utilizzato per installare l'SDK, è possibile includere l'SDK nel tuo codice con una sola istruzione `require`. Consulta la tabella riportata di seguito per individuare il codice PHP che meglio si adatta alle esigenze della tecnica di installazione. Sostituisci tutte le istanze di `/path/to/` con il percorso effettivo sul sistema.

Tecnica di installazione	Richiedere istruzione
Utilizzo di Composer	`require '/path/to/vendor/autoload.php';`
Utilizzo del file phar	`require '/path/to/aws.phar';`
Utilizzo di ZIP	`require '/path/to/aws-autoloader.php';`

In questo argomento, si presuppone il metodo di installazione di Composer. Se usi un metodo di installazione diverso, è possibile fare riferimento a questa sezione per trovare il codice `require` corretto da utilizzare.

Scrivi il codice

Copia e incolla il codice seguente in un nuovo file sorgente. Salvate e assegnate un nome al file`hello-s3.php`.

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

```
use Aws\S3\S3Client;


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

//Create a S3Client
$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";
}
```

Esecuzione del programma

Apri un prompt dei comandi per eseguire il tuo programma PHP. La sintassi di comando tipica per eseguire un programma PHP è:

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

Questo codice di esempio non utilizza argomenti. Per eseguire questo codice, inserisci quanto segue nel prompt dei comandi:

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

Passaggi successivi

Per testare molte altre operazioni di Amazon S3, consulta il AWS Code Examples Repository su. GitHub

Utilizzare AWS Cloud9 con AWS SDK per PHP

> **Note**
>
> AWS Cloud9 non è più disponibile per i nuovi clienti. I clienti esistenti di AWS Cloud9 possono continuare a utilizzare il servizio normalmente. Ulteriori informazioni.

AWS Cloud9 è un ambiente di sviluppo integrato (IDE) basato sul Web che contiene una raccolta di strumenti da utilizzare per codificare, creare, eseguire, testare, eseguire il debug e rilasciare software nel cloud. È possibile utilizzare AWS Cloud9 con AWS SDK per PHP per scrivere ed eseguire il codice PHP utilizzando un browser. AWS Cloud9 include strumenti come un editor di codice e un terminale. Poiché l' AWS Cloud9 IDE è basato sul cloud, puoi lavorare sui tuoi progetti dall'ufficio, da casa o ovunque utilizzando una macchina connessa a Internet. Per informazioni generali su AWS Cloud9, consulta la Guida per l'AWS Cloud9 utente.

Segui queste istruzioni per eseguire la configurazione AWS Cloud9 con AWS SDK per PHP:

- Passaggio 1: configura il tuo Account AWS da utilizzare AWS Cloud9
- Fase 2: Configura il tuo ambiente di AWS Cloud9 sviluppo
- Fase 3: Configurare il AWS SDK per PHP
- Passaggio 4: scarica il codice di esempio
- Passaggio 5: Esegui il codice di esempio

Passaggio 1: configura il tuo Account AWS da utilizzare AWS Cloud9

Per utilizzarlo AWS Cloud9, accedi alla AWS Cloud9 console da AWS Management Console.

> **Note**
>
> Se utilizzi AWS IAM Identity Center per autenticarti, potresti dover aggiungere l'autorizzazione richiesta di `iam:ListInstanceProfilesForRole` alla policy associata all'utente nella console IAM.

Per configurare un'entità IAM nel tuo AWS account per accedere AWS Cloud9 e accedere alla AWS Cloud9 console, consulta Team Setup AWS Cloud9 nella Guida per l'AWS Cloud9 utente.

Fase 2: configura il tuo ambiente di AWS Cloud9 sviluppo

Dopo aver effettuato l'accesso alla AWS Cloud9 console, utilizza la console per creare un ambiente di AWS Cloud9 sviluppo. Dopo aver creato l'ambiente, AWS Cloud9 apre l'IDE per quell'ambiente.

Per i dettagli, consulta Creazione di un ambiente AWS Cloud9 nella Guida AWS Cloud9 per l'utente.

> **Note**
>
> Quando crei il tuo ambiente nella console per la prima volta, ti consigliamo di scegliere l'opzione Create a new instance for environment (EC2). Questa opzione indica AWS Cloud9 di creare un ambiente, avviare un' EC2istanza Amazon e quindi connettere la nuova istanza al nuovo ambiente. Questo è il modo più veloce per iniziare a usare AWS Cloud9.

Apri il terminale se non è già aperto nell'IDE. Nella barra del menu nell'IDE, scegli Window, New Terminal (Finestra, Nuovo terminale). È possibile utilizzare la finestra del terminale per installare strumenti e creare applicazioni.

Fase 3: Configurare il AWS SDK per PHP

Dopo aver AWS Cloud9 aperto l'IDE per il tuo ambiente di sviluppo, usa la finestra del terminale per configurarlo AWS SDK per PHP nel tuo ambiente.

Composer è il metodo consigliato per installare. AWS SDK per PHP Composer è uno strumento per PHP in grado di gestire e installare le dipendenze del progetto.

Per ulteriori informazioni su come installare Composer, configurare il caricamento automatico e seguire le altre best practice per la definizione delle dipendenze, consulta getcomposer.org.

Installa Composer

Se Composer non è già presente nel progetto, scarica e installa Composer dalla pagina Scarica Composer.

- Per Windows, seguite le istruzioni di Windows Installer.
- Per Linux, segui le istruzioni di installazione della riga di comando.

Aggiungi AWS SDK per PHP come dipendenza tramite Composer

Se Composer è già installato globalmente sul sistema, esegui quanto segue nella directory di base del progetto per installarlo AWS SDK per PHP come dipendenza:

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

Altrimenti, digitate questo comando Composer per installare la versione più recente di AWS SDK per PHP come dipendenza.

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

Aggiungi l'autoloader ai tuoi script php

L'installazione di Composer crea diverse cartelle e file nell'ambiente. Il file principale che utilizzerai è `autoload.php`, che si trova nella cartella `vendor` dell'ambiente.

Per utilizzarli AWS SDK per PHP nei tuoi script, includi il caricatore automatico negli script, come segue.

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

Passaggio 4: Scarica il codice di esempio

Usa la finestra del terminale per scaricare il codice di esempio da utilizzare AWS SDK per PHP nell'ambiente di AWS Cloud9 sviluppo.

Per scaricare una copia di tutti gli esempi di codice utilizzati nella documentazione ufficiale dell' AWS SDK nella directory principale dell'ambiente, esegui il comando seguente:

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

Gli esempi di codice per si AWS SDK per PHP trovano nella `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` directory, dove `ENVIRONMENT_NAME` è riportato il nome dell'ambiente di sviluppo.

Per seguire l'utilizzo di un esempio di Amazon S3, consigliamo di iniziare con un esempio di codice. `ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php`

Questo esempio elencherà i tuoi bucket Amazon S3. Usa la finestra del terminale per accedere alla `s3` directory ed elencare i file.

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

Per aprire il file AWS Cloud9, è possibile fare clic `ListBuckets.php` direttamente nella finestra del terminale.

Per ulteriore assistenza nella comprensione degli esempi di codice, vedi Esempi di AWS SDK per PHP codice.

Passaggio 5: Esegui codice di esempio

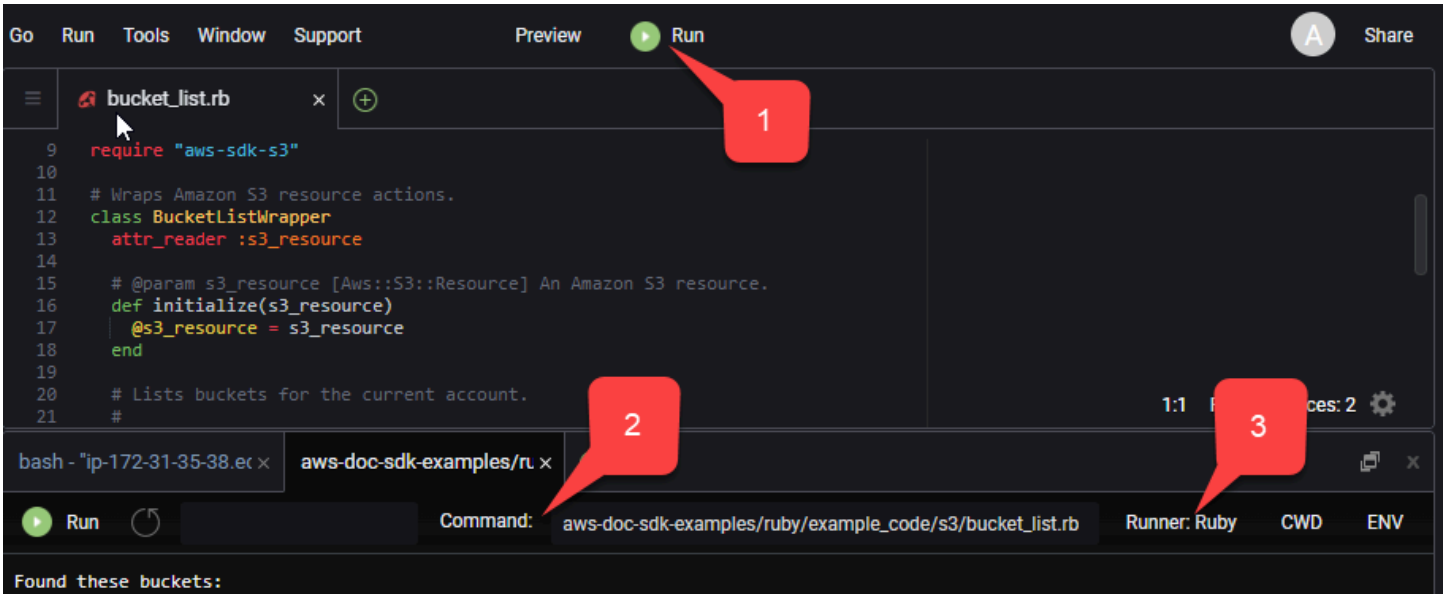
Per eseguire il codice nel tuo ambiente di AWS Cloud9 sviluppo, scegli il pulsante Esegui nella barra dei menu in alto. AWS Cloud9 rileva automaticamente l'estensione del `.php` file e utilizza il runner PHP (server web integrato) per eseguire il codice. Tuttavia, per questo esempio vogliamo effettivamente l'opzione PHP (). **`cli`** Per ulteriori informazioni sull'esecuzione del codice in AWS Cloud9, consulta Run Your Code nella Guida per l'AWS Cloud9 utente.

Nella schermata seguente, prendi nota di queste aree di base:

- 1: Esegui. Il pulsante Esegui si trova nella barra dei menu in alto. Si apre una nuova scheda per i risultati.

  > **Note**
  >
  > Puoi anche creare manualmente nuove configurazioni di esecuzione. Nella barra dei menu, selezionare Run (Esegui), Run Configurations (Configurazioni esecuzione), New Run Configuration (Nuova configurazione esecuzione).

- 2: Comando. AWS Cloud9 compila la casella di testo Comando con il percorso e il nome del file eseguito. Se il codice prevede che vengano passati parametri della riga di comando, questi possono essere aggiunti alla riga di comando nello stesso modo in cui si farebbe eseguendo il codice in una finestra di terminale.
- 3: Runner. AWS Cloud9 rileva che l'estensione del file è `.php` e seleziona il PHP (server web integrato) Runner per eseguire il codice. Seleziona invece PHP (**`cli`**) per eseguire questo esempio.



Qualsiasi output generato dal codice in esecuzione viene visualizzato nella scheda.

Configurare la AWS SDK per PHP versione 3

AWS SDK per PHP È costituito da varie funzionalità e componenti. Ciascuno dei seguenti argomenti descrive i componenti utilizzati nell'SDK.

La AWS SDKs and Tools Reference Guide contiene anche impostazioni, funzionalità e altri concetti fondamentali comuni a molti di AWS SDKs.

Argomenti

- Modelli di utilizzo di base della AWS SDK per PHP versione 3
- Configurazione per la AWS SDK per PHP versione 3
- Credenziali per la AWS SDK per PHP versione 3
- Oggetti di comando nella AWS SDK per PHP versione 3
- Promesse nella AWS SDK per PHP versione 3
- Gestori e middleware nella versione 3 AWS SDK per PHP
- Stream nella AWS SDK per PHP versione 3
- Paginatori nella versione 3 AWS SDK per PHP
- Camerieri nella versione 3 AWS SDK per PHP
- JMESPath espressioni nella AWS SDK per PHP versione 3
- Usa l'estensione AWS Common Runtime (AWS CRT)
- Aggiornamento dalla versione 2 di AWS SDK per PHP
- Condivisi config e file credentials
- Profili denominati

Modelli di utilizzo di base della AWS SDK per PHP versione 3

Questo argomento descrive i modelli di utilizzo di base dell' AWS SDK per PHP.

Prerequisiti

- Scarica e installa l'SDK
- Prima di utilizzare il AWS SDK per PHP, è necessario autenticarsi con. AWS Per informazioni sulla configurazione dell'autenticazione, vedere Autenticazione SDK con AWS

Inclusione dell'SDK nel codice

Qualsiasi tecnica abbia utilizzato per installare l'SDK, è possibile includere l'SDK nel tuo codice con una sola istruzione `require`. Consulta la tabella riportata di seguito per individuare il codice PHP che meglio si adatta alle esigenze della tecnica di installazione. Sostituisci tutte le istanze di `/path/to/` con il percorso effettivo sul sistema.

Tecnica di installazione	Richiedere istruzione
Utilizzo di Composer	`require '/path/to/vendor/autoload.php';`
Utilizzo del file phar	`require '/path/to/aws.phar';`
Utilizzo di ZIP	`require '/path/to/aws-autoloader.php';`

In questo argomento, si presuppone il metodo di installazione di Composer. Se usi un metodo di installazione diverso, è possibile fare riferimento a questa sezione per trovare il codice `require` corretto da utilizzare.

Riepilogo dell'utilizzo

Per utilizzare l'SDK per interagire con un AWS servizio, crea un'istanza di un oggetto Client. Gli oggetti client hanno metodi che corrispondono alle operazioni nell'API del servizio. Per eseguire una determinata operazione, è necessario scegliere il metodo corrispondente. Questo metodo restituisce un oggetto risultato del tipo array in caso di esito positivo oppure genera un'eccezione in caso di errore.

Creazione di un client

È possibile creare un client trasferendo un array associativo di opzioni a un costruttore di client.

Importazioni

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

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

Codice di esempio

```
//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'.
```

Le informazioni sul parametro opzionale «version» sono disponibili nell'argomento delle opzioni di configurazione.

Si noti che non abbiamo esplicitamente fornito le credenziali al client. Questo perché l'SDK utilizza la catena di provider di credenziali predefinita per cercare informazioni sulle credenziali.

Tutte le opzioni generali di configurazione del client sono descritte in dettaglio in. Configurazione per la AWS SDK per PHP versione 3 La gamma di opzioni fornite a un client può variare in base a quale client si sta creando. Queste opzioni personalizzate per la configurazione del client sono descritte nella documentazione API per ogni client.

Utilizzo della **Sdk** classe

La classe `Aws\Sdk` agisce come un client factory e viene utilizzata per gestire le opzioni di configurazione condivise da più client. Molte delle opzioni che possono essere fornite a un costruttore client specifico possono essere fornite anche alla `Aws\Sdk` classe. Queste opzioni vengono poi applicate a ciascun costruttore di client.

Importazioni

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

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

Codice di esempio

```
// 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();
```

Le opzioni che sono condivise per tutti i client sono collocate in coppie chiave-valore a livello radice. I dati di configurazione specifici del servizio possono essere forniti in una chiave che è la stessa del namespace di un servizio (ad esempio, «S3", «», ecc.). 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();
```

I valori di configurazione specifici del servizio sono una combinazione di valori specifici del servizio e di valori a livello radice (ad esempio, i valori specifici del servizio sono clonati in valori di livello radice).

> **Note**
>
> È consigliabile utilizzare la classe `Sdk` per creare client se nell'applicazione si utilizzano più istanze client. La classe `Sdk` utilizza automaticamente lo stesso client HTTP per ogni client SDK, consentendo a client SDK per servizi diversi di eseguire richieste HTTP senza blocchi. Se i client SDK non utilizzano lo stesso client HTTP, le richieste HTTP inviate dal client SDK potrebbero bloccare l'orchestrazione della promessa tra i servizi.

Esecuzione delle operazioni di servizio

È possibile eseguire un'operazione di servizio chiamando il metodo con lo stesso nome di un oggetto client. Ad esempio, per eseguire l'PutObjectoperazione Amazon S3, è necessario chiamare il `Aws\S3\S3Client::putObject()` metodo.

Importazioni

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

use Aws\S3\S3Client;
```

Codice di esempio

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

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

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

Le operazioni disponibili per un client e la struttura di input e output sono definite in fase di runtime in base a un file di descrizione del servizio. Durante la creazione di un client, è necessario fornire una versione (ad esempio, "2006-03-01" o "più recente"). L'SDK individua il file di configurazione corrispondente in base alla versione fornita.

Tutti i metodi di operazione come `putObject()` accettano un unico argomento, un array associativo che indica i parametri dell'operazione. La struttura di questo array (e la struttura dell'oggetto risultato) è definita per ciascuna operazione nella documentazione API SDK (ad esempio, consulta la documentazione API per l'operazione putObject).

Opzioni del gestore HTTP

È inoltre possibile ottimizzare il modo in cui il gestore HTTP sottostante esegue la richiesta utilizzando il parametro speciale `@http`. Le opzioni che possono essere incluse nel parametro `@http` sono le stesse che possono essere impostate quando si crea un'istanza del client con l'opzione client "http".

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

Richieste asincrone

È possibile inviare comandi simultaneamente utilizzando le funzionalità asincrone dell'SDK. È possibile inviare le richieste in modo asincrono aggiungendo un suffisso al nome dell'operazione con `Async`. In questo modo si avvia la richiesta e si restituisce una promessa. La promessa viene soddisfatta con l'oggetto risultato in caso di esito positivo o rifiutata con un'eccezione in caso di errore. In questo modo è possibile creare più promesse e richiedere l'invio simultaneo di richieste HTTP quando il gestore HTTP sottostante trasferisce le richieste.

Importazioni

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

Codice di esempio

```
// 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();
```

È possibile forzare il completamento sincrono di una promessa utilizzando il metodo `wait` della promessa. Forzando il completamento della promessa, inoltre, "si apre" lo stato della promessa per impostazione predefinita, ovvero viene restituito l'esito della promessa oppure viene generata l'eccezione che è stata riscontrata. Quando si richiama `wait()` su una promessa, il processo si blocca fino a quando la richiesta HTTP non viene completata e il risultato viene popolato, oppure viene creata un'eccezione.

Quando si utilizza l'SDK con una libreria di loop eventi, non bloccare i risultati. Occorre utilizzare invece il metodo `then()` di un risultato per accedere a una promessa che viene risolta o rifiutata al termine dell'operazione.

Importazioni

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

Codice di esempio

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

Lavorare con oggetti di risultato

L'esecuzione di un'operazione corretta restituisce un oggetto `Aws\Result`. Anziché restituire i dati grezzi XML o JSON di un servizio, l'SDK forza i dati di risposta in una struttura array associativa. Normalizza alcuni aspetti dei dati in base alla sua conoscenza del servizio specifico e la struttura di risposta sottostante.

È possibile accedere ai dati dall' AWSResult oggetto come un array PHP associativo.

Importazioni

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

Codice di esempio

```
// 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();
```

Il contenuto dell'oggetto risultato dipende dall'operazione che è stata eseguita e dalla versione di un servizio. La struttura che risulta da ogni operazione API è documentata nella documentazione API per ciascuna operazione.

L'SDK è integrato con JMESPathun DSL utilizzato per cercare e manipolare dati JSON o, nel nostro caso, array PHP. L'oggetto risultato contiene un metodo `search()` che è possibile utilizzare per estrarre dati dal risultato in modo più dichiarativo.

Codice di esempio

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

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

Gestione degli errori

Gestione degli errori sincroni

Se si verifica un errore durante l'esecuzione di un'operazione, viene generata un'eccezione. Per questo motivo, se è necessario gestire gli errori nel codice, utilizza blocchi `try/catch` intorno alle operazioni. L'SDK genera eccezioni specifiche per il servizio quando si verifica un errore.

Gli esempi seguenti utilizzano `Aws\S3\S3Client`. Se si verifica un errore, l'eccezione generata sarà del tipo `Aws\S3\Exception\S3Exception`. Tutte le eccezioni specifiche per il servizio generate dall'SDK si estendono dalla classe `Aws\Exception\AwsException`. Questa classe contiene informazioni utili sull'errore, tra cui l'id della richiesta, il codice errore e il tipo di errore. Nota per alcuni servizi che la supportano, i dati di risposta vengono convertiti in una struttura array associativa (simile agli oggetti `Aws\Result`), a cui è possibile accedere come un normale array associativo PHP. Il metodo `toArray()` restituirà tali dati, se esistenti.

Importazioni

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

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

Codice di esempio

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

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

Gestione asincrona degli errori

Le eccezioni non vengono generate durante l'invio di richieste asincrone. Al contrario, è necessario utilizzare il metodo `then()` o `otherwise()` della promessa restituita per ricevere il risultato o l'errore.

Importazioni

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

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

Codice di esempio

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

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

È possibile "aprire" la promessa e causare invece la creazione dell'eccezione.

Importazioni

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

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

Codice di esempio

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

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

Configurazione per la AWS SDK per PHP versione 3

Le opzioni del costruttore client possono essere fornite in un costruttore client o fornite alla `Aws\Sdk`classe. La vasta gamma di opzioni fornite a uno specifico tipo di client può variare in base

a quale client si sta creando. Queste opzioni personalizzate per la configurazione di client sono descritte nella documentazione API di ogni client.

Nota che alcune opzioni di configurazione controlleranno e utilizzeranno i valori predefiniti basati su variabili di ambiente o su un AWS file di configurazione. Per impostazione predefinita, il file di configurazione da controllare sarà `.aws/config` nella home directory, comunemente `~/.aws/config`. Tuttavia, è possibile utilizzare la variabile d'ambiente `AWS_CONFIG_FILE` per impostare la posizione del file di configurazione predefinito. Ad esempio, questo può essere utile se si limita l'accesso ai file a determinate directory con. `open_basedir`

Per ulteriori informazioni sulla posizione e sulla formattazione dei `credentials` file condivisi e condivisi, consulta Configuration nella AWS `config` AWS SDKs and Tools Reference Guide.

Per i dettagli su tutte le impostazioni di configurazione globali che è possibile impostare nei file di AWS configurazione o come variabili di ambiente, consulta il riferimento alle impostazioni di configurazione e autenticazione nella Guida di riferimento AWS SDKs e agli strumenti.

Opzioni di configurazione

- api_provider
- credenziali
- debug
- statistiche
- endpoint
- endpoint_provider
- endpoint_discovery
- handler
- http
- http_handler
- profilo
- Regione
- retries
- scheme
- service
- signature_provider

- signature_version
- ua_append
- use_aws_shared_config_files
- validate
- version

L'esempio seguente mostra come passare opzioni a un costruttore di client Amazon S3.

```
use Aws\S3\S3Client;

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

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

Consulta la guida di utilizzo di base per ulteriori informazioni su come costruire client.

api_provider

Tipo

`callable`

Un PHP chiamabile che accetta un tipo, servizio e argomento versione e restituisce un array di dati di configurazione corrispondenti. Il valore del tipo può essere `api`, `waiter` o `paginator`.

Per impostazione predefinita, l'SDK usa un'istanza di `Aws\Api\FileSystemApiProvider` che carica i file dell'API dalla cartella `src/data` dell'SDK.

credenziali

Tipo

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

Passa un oggetto `Aws\Credentials\CredentialsInterface` per utilizzare le credenziali di un'istanza specifica. Quanto segue specifica che deve essere utilizzato il provider di credenziali IAM Identity Center. Questo provider è noto anche come provider di credenziali SSO.

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

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

Se utilizzi un profilo denominato, sostituisci il nome del tuo profilo con '`default`' nell'esempio precedente. Per ulteriori informazioni sulla configurazione di profili denominati, consulta `configShared and credentials files` nella AWS SDKs and Tools Reference Guide.

Se non specificate un provider di credenziali da utilizzare e vi affidate alla catena di fornitori di credenziali, il messaggio di errore derivante dall'autenticazione non riuscita è in genere generico. Viene generato dall'ultimo provider nell'elenco delle fonti controllate per verificare la presenza di credenziali valide, che potrebbe non essere il provider che si sta tentando di utilizzare. Quando si specifica il provider di credenziali da utilizzare, qualsiasi messaggio di errore risultante è più utile e pertinente perché proviene solo da quel provider. Per ulteriori informazioni sulla catena di fonti in cui è stata verificata la presenza di credenziali, consulta Credential provider chain nella AWS SDKs and Tools Reference Guide.

Passa `false` per utilizzare credenziali null e non firmare richieste.

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

Passa una funzione di provider di credenziali chiamabile per creare credenziali utilizzando una funzione.

```
use Aws\Credentials\CredentialProvider;

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

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

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

Trasferisce le credenziali nella cache a un'istanza di `Aws\CacheInterface` per memorizzare nella cache i valori restituiti dalla catena di provider predefinita su più processi.

```
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
]);
```

Puoi trovare ulteriori informazioni sulla fornitura di credenziali a un client nella guida Credentials for the AWS SDK per PHP Version 3.

> **Note**
>
> Quando vengono utilizzate, le credenziali vengono caricate e convalidate lentamente.

debug

Tipo

`bool|array`

Emette le informazioni di debug su ogni trasferimento. Le informazioni di debug contengono informazioni su ogni modifica dello stato di una transazione e vengono preparate e inviate tramite la rete. L'output di debug include anche informazioni sul gestore HTTP specifico utilizzato da un client (ad esempio, debug cURL output).

Imposta su `true` per visualizzare le informazioni di debug durante l'invio delle richieste.

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

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

In alternativa, puoi fornire un array associativo con le seguenti chiavi.

logfn (callable)

Funzione richiamata con messaggi di log. Per impostazione predefinita, viene utilizzata la funzione `echo` di PHP.

stream_size (int)

Quando le dimensioni di un flusso sono maggiori rispetto a questo numero, il flusso di dati non viene registrato. Imposta su `0` per non registrare alcun dato di flusso.

scrub_auth (bool)

Imposta per `false` disabilitare la cancellazione dei dati di autenticazione dai messaggi registrati (il che significa che l'ID della chiave di AWS accesso e la firma verranno trasmessi a). `logfn`

http (bool)

Imposta su `false` per disattivare la funzione di "debug" dei gestori HTTP di livello inferiore (ad esempio, verbose cURL output).

auth_headers (array)

Imposta su una mappatura chiave-valore di intestazioni che desideri sostituire mappate al valore con cui desideri sostituirle. Questi valori non vengono utilizzati a meno che `scrub_auth` non sia impostato su `true`.

auth_strings (array)

Imposta su una mappatura chiave-valore di espressioni regolari da mappare alle loro sostituzioni. Questi valori sono utilizzati dallo scrubber dei dati di autenticazione se `scrub_auth` è impostato su `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**
>
> Questa opzione restituisce anche le informazioni sottostanti sul gestore HTTP prodotte dall'opzione debug. `http` L'output di debug è molto utile durante la diagnostica dei problemi nell' AWS SDK per PHP. Fornisci l'output di debug per un caso di errore isolato all'apertura dei problemi sull'SDK.

statistiche

Tipo

`bool|array`

Associa le statistiche di trasferimento agli errori e ai risultati restituiti dalle operazioni dell'SDK.

Imposta su `true` per raccogliere le statistiche di trasferimento sulle richieste inviate.

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

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

In alternativa, puoi fornire un array associativo con le seguenti chiavi.

retries (bool)

Impostata per `false` disabilitare la segnalazione dei tentativi di tentativo. Le statistiche sui nuovi tentativi sono raccolte per impostazione predefinita e restituite.

http (bool)

Impostato `true` per consentire la raccolta di statistiche da adattatori HTTP di livello inferiore (ad esempio, valori restituiti). GuzzleHttpTransferStats I gestori HTTP devono supportare un'opzione __on_transfer_stats affinché questo abbia effetto. Le statistiche HTTP vengono restituite come un array indicizzato di array associativi; ogni array associativo contiene le statistiche di trasferimento restituite per una richiesta dal gestore HTTP del client. Disabilitato per impostazione predefinita.

Se una richiesta è stata tentata nuovamente, ne vengono restituite tutte le statistiche di trasferimento, con `$result['@metadata']['transferStats']['http'][0]` contenenti le statistiche per la prima richiesta, `$result['@metadata']['transferStats']['http'][1]` che contengono le statistiche per la seconda richiesta e così via.

timer (bool)

Imposta su `true` per abilitare un timer di comando che indica il tempo totale utilizzato, in secondi, per eseguire un'operazione. Disabilitato per impostazione predefinita.

```
$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'];
```

endpoint

Tipo

```
string
```

L'URI completo del servizio Web. Ciò è necessario per i servizi, ad esempio quelli che utilizzano AWS Elemental MediaConvertendpoint specifici dell'account. Per questi servizi, richiedi questo endpoint utilizzando il metodo. `describeEndpoints`

Questo è necessario solo quando ci si connette a un endpoint personalizzato (ad esempio, una versione locale di Amazon S3 o Amazon DynamoDB Local).

Ecco un esempio di connessione ad Amazon DynamoDB Local:

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

Consulta le AWS regioni e gli endpoint per un elenco delle AWS regioni e degli endpoint disponibili.

endpoint_provider

Tipo

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

Un'istanza opzionale di EndpointProvider V2 o PHP callable che accetta un hash di opzioni, tra cui una chiave «service» e «region». Restituisce NULL o un hash di dati di endpoint, di cui la chiave "endpoint" è obbligatoria.

Ecco un esempio di come creare un provider di endpoint minimi.

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

endpoint_discovery

Tipo

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

Il rilevamento endpoint identifica e si connette all'endpoint corretto per un'API di servizio che supporta il rilevamento di endpoint. Per servizi che supportano ma non richiedono il rilevamento di endpoint, abilita `endpoint_discovery` durante la creazione client. Se un servizio non supporta il rilevamento di endpoint questa configurazione viene ignorata.

`Aws\EndpointDiscovery\ConfigurationInterface`

Un provider di configurazione opzionale che abilita la connessione automatica all'endpoint appropriato di un'API di servizio per operazioni specificate dal servizio.

L'oggetto `Aws\EndpointDiscovery\Configuration` accetta due opzioni, incluso un valore booleano, "abilitato", che indica se il rilevamento di endpoint è abilitato e un "cache_limit" intero che indica il numero massimo di chiavi nella cache di endpoint.

Per ogni client creato, passa un oggetto `Aws\EndpointDiscovery\Configuration` per utilizzare una configurazione specifica per il rilevamento di endpoint.

```
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,

]);
```

Passa un'istanza di `Aws\CacheInterface` per memorizzare nella cache i valori restituiti dal rilevamento di endpoint su più processi.

```
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),
]);
```

Passa un array al rilevamento di endpoint.

```
use Aws\S3\S3Client;

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

handler

Tipo

`callable`

Un gestore che accetta un oggetto di comando e un oggetto di richiesta e che restituisce una promessa (`GuzzleHttp\Promise\PromiseInterface`) che viene soddisfatta con un oggetto `Aws`

`\ResultInterface` o rifiutata con un `Aws\Exception\AwsException`. Un gestore non accetta un gestore successivo poiché è terminale e tenuto a soddisfare un comando. Se non viene fornito alcun gestore, viene utilizzato un gestore Guzzle predefinito.

Puoi usare `Aws\MockHandler` per restituire risultati fittizi o generare eccezioni fittizie. Metti in coda i risultati o le eccezioni e li rimuoverà dalla MockHandler coda in ordine 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

Tipo

`array`

Impostato su una gamma di opzioni di HTTP che vengono applicate alle richieste e ai trasferimenti HTTP creati tramite l'SDK.

L'SDK supporta le opzioni di configurazione seguenti:

cert

Tipo

```
string|array
```

Specifica il certificato lato client formattato PEM.

- Imposta come stringa per il percorso del solo file di certificato.

```
use Aws\S3\S3Client;

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

- Imposta come array contenente il percorso e la password.

```
use Aws\S3\S3Client;

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

connect_timeout

Un float che descrive il numero di secondi da attendere durante il tentativo di connettersi a un server. Utilizza `0` per attendere un periodo di tempo indefinito (comportamento predefinito).

```
use Aws\DynamoDb\DynamoDbClient;
```

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

debug

Tipo

`bool|resource`

Istruisce il gestore HTTP sottostante di emettere le informazioni di debug. Le informazioni di debug fornite da diversi gestori HTTP possono variare.

- Passa `true` per scrivere l'output di debug in STDOUT.
- Passa una `resource` come restituita da `fopen` per scrivere l'output di debug in un determinato flusso di risorse PHP.

decode_content

Tipo

`bool`

Istruisce il gestore HTTP sottostante di decomprimere il corpo delle risposte compromesse. Quando non è abilitato, i corpi compressi delle risposte potrebbero essere decompressi con un `GuzzleHttp\Psr7\InflateStream`.

> **Note**
>
> La decodifica dei contenuti è abilitata per impostazione predefinita nel gestore HTTP predefinito dell'SDK. Per garantire la compatibilità con le versioni precedenti, questa impostazione predefinita non può essere modificata. Se archivi file compressi in Amazon S3, ti consigliamo di disabilitare la decodifica dei contenuti a livello di client S3.

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

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

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

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

delay

Tipo

`int`

Il numero di ritardo in millisecondi prima dell'invio della richiesta. Questo è spesso usato per inserire un ritardo prima di inviare di nuovo la richiesta.

expect

Tipo

`bool|string`

Questa opzione viene passata tramite il gestore HTTP sottostante. Per impostazione predefinita, l'intestazione Expect: 100-Continue è impostata quando il corpo della richiesta supera 1 MB. `true` o `false` abilita o disabilita l'intestazione su tutte le richieste. Se viene utilizzato un intero, solo le richieste con corpi che superano questa impostazione utilizzeranno l'intestazione. Quando utilizzato come un intero, se le dimensioni del corpo non sono note, verrà inviata l'intestazione Expect.

> **Warning**
>
> La disabilitazione dell'intestazione Expect può impedire al servizio da restituire l'autenticazione o altri errori. Questa opzione deve essere configurata con cautela.

progress

Tipo

`callable`

Definisce una funzione da invocare quando vi è un avanzamento del trasferimento. La funzione accetta i seguenti argomenti:

1. Il numero totale di byte previsti che saranno scaricati.
2. Il numero di byte scaricati fino a ora.
3. Il numero totale di byte previsti che saranno caricati.
4. Il numero di byte caricati fino a ora.

```
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

Tipo

```
string|array
```

Puoi connetterti a un AWS servizio tramite un proxy utilizzando l'opzione. `proxy`

- Fornisci un valore di stringa per connetterti a un proxy per tutti i tipi di URIs. Il valore di stringa proxy può contenere uno schema, nome utente e password. Ad esempio "`http://username:password@192.168.16.1:10`".
- Fornisci un array associativo delle impostazioni proxy in cui la chiave è la combinazione di URI, e il valore è il proxy per un determinato URI (ad esempio, puoi fornire diversi proxy per endpoint "http" e "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',
        ]
    ]
]);
```

Puoi utilizzare la variabile ambiente `HTTP_PROXY` per configurare un proxy specifico del protocollo "http" e la variabile ambiente `HTTPS_PROXY` per configurare un proxy specifico "https".

sink

Tipo

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

L'opzione `sink` controlla dove vengono scaricati i dati di risposta di un'operazione.

- Fornisci un `resource` come restituito da `fopen` per scaricare il corpo della risposta in un flusso PHP.
- Fornisci il percorso di un file su disco come un valore `string` per scaricare il corpo della risposta su un determinato file su disco.
- Fornisci un `Psr\Http\Message\StreamInterface` per scaricare il corpo della risposta in un oggetto di flusso PSR specifico.

> **Note**
>
> L'SDK scarica il corpo della risposta in un flusso temp PHP per impostazione predefinita. Ciò significa che i dati rimangono in memoria fino a quando le dimensioni del corpo non raggiungono 2 MB; dopodiché i dati vengono scritti in un file temporaneo su disco.

synchronous

Tipo

```
bool
```

L'opzione `synchronous` informa il gestore HTTP che desideri bloccare il risultato.

stream

Tipo

```
bool
```

Imposta su `true` per comunicare al gestore HTTP sottostante che desideri trasmettere il corpo di una risposta dal servizio Web, anziché scaricarlo tutto subito. Ad esempio, questa opzione viene utilizzata nella classe stream wrapper di Amazon S3 per garantire lo streaming dei dati.

timeout

Tipo

`float`

Un float che descrive il timeout della richiesta in secondi. Utilizza `0` per attendere un periodo di tempo indefinito (comportamento predefinito).

```
use Aws\DynamoDb\DynamoDbClient;

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

Verifica

Tipo

`bool|string`

Puoi personalizzare il comportamento di verifica del certificato SSL/TLS dell'SDK tramite l'opzione `verify` `http`.

- Imposta su `true` per abilitare la verifica del certificato SSL/TLS peer e utilizzare il bundle CA predefinito fornito dal sistema operativo.
- Imposta su `false` per disabilitare la verifica del certificato peer. (Questa operazione non è sicura!)
- Imposta su una stringa per fornire il percorso a un bundle di certificazione CA per abilitare la verifica utilizzando un bundle CA personalizzato.

Se il bundle CA non viene individuato per il sistema e ricevi un errore, specifica il percorso di un bundle CA per l'SDK. Se non hai bisogno di un bundle CA specifico, Mozilla fornisce un bundle CA comune che puoi scaricare qui (viene gestito dal servizio maintainer di cURL). Una volta che disponi di un bundle CA su disco, puoi configurare l'impostazione.ini PHP `openssl.cafile` per puntare al percorso del file, consentendo di omettere l'opzione di richiesta `verify`. Per ulteriori dettagli sui certificati SSL, consulta il sito Web di 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

Tipo

```
callable
```

L'opzione `http_handler` viene utilizzata per integrare l'SDK con altri client HTTP. Un'opzione `http_handler` è una funzione che accetta un oggetto `Psr\Http\Message\RequestInterface` e una vasta gamma di opzioni `http` applicate al comando e restituisce un oggetto `GuzzleHttp\Promise\PromiseInterface` che viene soddisfatto con un oggetto `Psr\Http\Message\ResponseInterface` o rifiutato con una vasta gamma dei seguenti dati di eccezione:

- `exception` - (`\Exception`) l'eccezione che è stata riscontrata.
- `response` - (`Psr\Http\Message\ResponseInterface`) la risposta che è stata ricevuta (se presente).

- `connection_error` - (bool) imposta su `true` per segnalare l'errore come un errore di connessione. L'impostazione di questo valore su `true` consente anche all'SDK di effettuare nuovamente l'operazione, se necessario.

L'SDK converte automaticamente l'`http_handler` dato in una normale opzione `handler` includendo l'`http_handler` fornito con un oggetto `Aws\WrappedHttpHandler`.

Per impostazione predefinita, l'SDK utilizza Guzzle come gestore HTTP. È possibile fornire qui un gestore HTTP diverso o fornire un client Guzzle con le proprie opzioni personalizzate definite.

Impostazione della versione TLS

Un caso d'uso è quello di impostare la versione TLS utilizzata da Guzzle con Curl, supponendo che Curl sia installato nel proprio ambiente. Notare i vincoli di versione Curl per cui la versione di TLS è supportata. Per impostazione predefinita viene utilizzata la versione più recente. Se la versione TLS è impostata in modo esplicito e il server remoto non supporta questa versione, verrà generato un errore anziché utilizzare una versione TLS precedente.

È possibile individuare la versione TLS utilizzata per una determinata operazione client impostando l'opzione client `debug` su true ed esaminando l'output della connessione SSL. Quella linea potrebbe essere simile a `SSL connection using TLSv1.2`

Esempio di impostazione di TLS 1.2 con Guzzle 6:

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

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

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

> **Note**
>
> Questa opzione `http_handler` sostituisce qualsiasi opzione `handler` fornita.

profilo

Tipo

`string`

L'opzione «profile» specifica quale profilo utilizzare quando le credenziali vengono create dal file delle AWS credenziali nella directory HOME (in genere). `~/.aws/credentials` Questa impostazione sostituisce la variabile ambiente `AWS_PROFILE`.

> **Note**
>
> Quando si specifica l'opzione «profilo», l'`"credentials"`opzione viene ignorata e le impostazioni relative alle credenziali nel file di AWS configurazione (in genere) vengono ignorate. `~/.aws/config`

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

Vedi Credenziali per la AWS SDK per PHP versione 3 per ulteriori informazioni sulla configurazione delle credenziali e sul formato di file.ini.

Regione

Tipo

`string`

Richiesto

true

AWS Regione a cui connettersi. Consulta le AWS regioni e gli endpoint per un elenco delle regioni disponibili.

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

retries

Tipo

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

Predefinita

`int(3)`

Configura la modalità nuovi tentativi e il numero massimo di tentativi consentiti per un client. Passa `0` per disabilitare i nuovi tentativi.

Le tre modalità di riprova sono:

- `legacy`- l'implementazione predefinita dei tentativi precedenti
- `standard`- aggiunge un sistema di quote di tentativi per evitare tentativi che difficilmente avranno successo
- `adaptive`: si basa sulla modalità standard, aggiungendo un limitatore di velocità lato client. Questa modalità è considerata sperimentale.

La configurazione dei nuovi tentativi consiste nella modalità e nel massimo tentativo da utilizzare per ogni richiesta. La configurazione può essere impostata in un paio di posizioni diverse, nel seguente ordine di precedenza.

Ordine di precedenza

L'ordine di precedenza per la configurazione dei tentativi è il seguente (1 sostituisce 2-3, ecc.):

1. Opzione di configurazione client
2. Variabili di ambiente
3. AWS File di configurazione condiviso

Variabili di ambiente

- `AWS_RETRY_MODE` impostato su `legacy`, `standard` o `adaptive`
- `AWS_MAX_ATTEMPTS` - impostato su un valore intero per il massimo di tentativi per richiesta

Chiavi file di configurazione condiviso

- `retry_mode` impostato su `legacy`, `standard` o `adaptive`
- `max_attempts` - impostato su un valore intero per il massimo di tentativi per richiesta

Configurazione del client

L'esempio seguente disabilita i nuovi tentativi per il client Amazon DynamoDB.

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

L'esempio seguente passa in un numero intero, che per impostazione predefinita sarà la modalità `legacy` con il numero passato di nuovi tentativi

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

L'oggetto **`Aws\Retry\Configuration`** accetta due parametri, la modalità di nuovi tentativi

e un numero intero per il numero massimo di tentativi per richiesta. Questo esempio passa in un

oggetto `Aws\Retry\Configuration` per ripetere la configurazione.

```
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,
]);
```

Questo esempio passa in un array per la configurazione dei nuovi tentativi.

```
use Aws\S3\S3Client;

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

In questo esempio viene passata un'istanza di `Aws\CacheInterface` per memorizzare nella cache i valori restituiti dal provider di configurazione dei nuovi tentativi predefinito.

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

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

```
]);
```

scheme

Tipo

`string`

Predefinita

`string(5) "https"`

Schema di URI da usare per la connessione. L'SDK usa endpoint "https" (ovvero, utilizza le connessioni SSL/TLS) per impostazione predefinita. Puoi tentare di connetterti a un servizio su un endpoint "http" non crittografato impostando `scheme` su "http".

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

Consulta le AWS regioni e gli endpoint per un elenco degli endpoint e per sapere se un servizio supporta lo schema. `http`

service

Tipo

`string`

Richiesto

true

Nome del servizio da utilizzare. Questo valore viene fornito per impostazione predefinita quando si usa un client fornito dall'SDK (ad esempio, `Aws\S3\S3Client`). Questa opzione è utile quando si testa un servizio non ancora pubblicato nell'SDK, ma che hai disponibile su disco.

signature_provider

Tipo

`callable`

Un callable che accetta un nome di versione della firma (ad esempio`v4`), un nome di servizio e una AWS regione e restituisce un `Aws\Signature\SignatureInterface` oggetto o `NULL` se il provider è in grado di creare un firmatario per i parametri specificati. Questo provider viene utilizzato per creare firmatari utilizzati dal client.

Esistono varie funzioni fornite dall'SDK nella classe `Aws\Signature\SignatureProvider` che possono essere utilizzati per creare provider di firma personalizzati.

signature_version

Tipo

`string`

Una stringa che rappresenta una versione di firma personalizzata per l'uso con un servizio (ad esempio `v4`, ecc.). In base alla firma dell'operazione, la versione POTREBBE ignorare questa versione di firma richiesta, se necessario.

Gli esempi seguenti mostrano come configurare un client Amazon S3 per utilizzare la versione 4 della firma:

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

> **Note**
>
> Il `signature_provider` utilizzato dal client DEVE essere in grado di creare l'opzione `signature_version` che fornisci. Il `signature_provider` predefinito utilizzato dall'SDK può creare oggetti di firma per le versioni di firma "v4" e "anonime".

ua_append

Tipo

```
string|string[]
```

Predefinita

```
[]
```

Una stringa o matrice di stringhe che vengono aggiunte alla stringa utente-agente passata al gestore HTTP.

use_aws_shared_config_files

Tipo

```
bool|array
```

Predefinita

```
bool(true)
```

Imposta su false per disabilitare il controllo del file di configurazione condiviso in '~/. aws/config' and '~/.aws/credentials'. Ciò sovrascriverà la variabile di AWS_CONFIG_FILE ambiente.

validate

Tipo

```
bool|array
```

Predefinita

```
bool(true)
```

Imposta su `false` per disabilitare la convalida del parametro lato client. Puoi notare che la disattivazione della convalida può migliorare leggermente le prestazioni del client, ma la differenza è trascurabile.

```
// Disable client-side validation
$s3 = new Aws\S3\S3Client([
```

```
    'version'  => '2006-03-01',
    'region'   => 'eu-west-1',
    'validate' => false
]);
```

Imposta su un array associativo di opzioni di convalida per abilitare vincoli di convalida specifici:

- `required` - Convalida che i parametri necessari sono presenti (attiva per impostazione predefinita).
- `min` - Convalida la lunghezza minima di un valore (attiva per impostazione predefinita).
- `max` - Convalida la lunghezza massima di un valore.
- `pattern` - Convalida che il valore corrisponde a un'espressione regolare.

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

version

Tipo

`string`

Richiesto

false

Questa opzione specifica la versione del servizio Web da utilizzare (ad esempio,`2006-03-01`).

A partire dalla versione 3.277.10 dell'SDK, l'opzione «versione» non è richiesta. Se non specifichi l'opzione «versione», l'SDK utilizza la versione più recente del client di servizio.

Due situazioni richiedono un parametro «versione» quando si crea un client di servizio.

- Si utilizza una versione dell'SDK PHP precedente alla 3.277.10.
- Utilizzi la versione 3.277.10 o successiva e desideri utilizzare una versione diversa dalla «più recente» per un client di servizio.

Ad esempio, lo snippet seguente utilizza la versione 3.279.7 dell'SDK, ma non la versione più recente di. `Ec2Client`

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

Specificando un vincolo di versione si garantisce che il codice non sarà interessato da un'interruzione delle modifiche apportate al servizio.

Puoi trovare un elenco delle versioni dell'API disponibili nella pagina della documentazione API di ciascun client. Se non riesci a caricare una versione dell'API specifica, potrebbe essere necessario aggiornare la copia dell'SDK.

Credenziali per la AWS SDK per PHP versione 3

Per informazioni di riferimento sui meccanismi di credenziali disponibili per AWS SDKs, vedere Credenziali e accesso nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

> **Important**
>
> Per motivi di sicurezza, si consiglia vivamente di non utilizzare l'account root per AWS l'accesso. Fai sempre riferimento alle best practice di sicurezza di IAM nella IAM User Guide per le raccomandazioni di sicurezza più recenti.

Precedenza delle impostazioni

Quando inizializzate un nuovo client di servizio senza fornire alcun argomento relativo alle credenziali, l'SDK utilizza la catena di provider di credenziali predefinita per trovare le credenziali. AWS L'SDK utilizza il primo provider nella catena che restituisce credenziali senza errori.

AWS SDK per PHP Ha una serie di posizioni che controlla per trovare i valori per le impostazioni globali e i fornitori di credenziali. Di seguito è riportato l'ordine di precedenza:

1. Qualsiasi impostazione esplicita impostata nel codice o su un client di servizio stesso ha la precedenza su qualsiasi altra cosa.
2. Utilizza credenziali dalle variabili di ambiente.

L'impostazione delle variabili di ambiente è utile se stai eseguendo lavori di sviluppo su una macchina diversa da un' EC2 istanza Amazon.

3. Condivisi `config` e file `credentials`.

   Questi sono gli stessi file utilizzati da altri SDKs e da AWS CLI.

Argomenti

- Collabora con i fornitori di credenziali
- Assumi un ruolo IAM
- Usa credenziali temporanee da AWS STS
- Crea clienti anonimi

Collabora con i fornitori di credenziali

Un provider di credenziali è una funzione che restituisce un valore `GuzzleHttp\Promise\PromiseInterface` che viene soddisfatto con un'istanza `Aws\Credentials\CredentialsInterface` o rifiutato con un valore `Aws\Exception\CredentialsException`. L'SDK fornisce diverse implementazioni delle funzioni del fornitore di credenziali oppure è possibile implementare una logica personalizzata per la creazione di credenziali o per ottimizzare il caricamento delle credenziali.

I provider di credenziali vengono indicati nell'opzione di costruzione del client `credentials`. I provider di credenziali sono asincroni, il che li costringe a essere valutati in modo pigro a ogni chiamata di un'operazione API. Per questo motivo, trasferire una funzione di un provider di credenziali a un costruttore di client SDK non comporta l'immediata convalida delle credenziali. Se il provider di credenziali non restituisce un oggetto credenziali, l'operazione API sarà respinta con un valore `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
]);
```

Argomenti

- Comprendi la catena di provider di credenziali predefinita
- Provider integrati nell'SDK
- Concatenamento dei fornitori
- Creazione di un provider personalizzato
- Memorizzazione delle credenziali

Comprendi la catena di provider di credenziali predefinita

La catena di provider di credenziali predefinita è composta da una serie di provider di credenziali integrati richiamati dall'SDK. È implementato dalla funzione del fornitore di credenziali DefaultProvider senza parametri. Dopo aver trovato credenziali valide, la ricerca viene interrotta.

AWS SDK per PHP Esegue i fornitori di credenziali nel seguente ordine:

- **`env`**provider: l'SDK cerca le chiavi di AWS accesso che sono state impostate come variabili di ambiente.
- **`assumeRoleWithWebIdentityCredentialProvider`**provider: l'SDK cerca le impostazioni dei file del ruolo IAM e del token di identità web.
- A questo punto della catena, l'SDK cerca la configurazione nei file condivisi AWS `config` e nei file. `credentials` L'SDK cerca la configurazione nel profilo «predefinito», ma se la variabile di `AWS_PROFILE` ambiente è impostata, l'SDK utilizza il valore del profilo denominato.
  - **`sso`**provider: l'SDK cerca le impostazioni di configurazione di IAM Identity Center nel file condiviso. `config`
  - **`process`**provider: l'SDK cerca l'`credential_process`impostazione nel file condivisocredentials.
  - **`ini`**provider: l'SDK cerca AWS le credenziali o le informazioni sul ruolo IAM nel file condiviso. `credentials`
  - **`process`**provider: l'SDK cerca l'`credential_process`impostazione nel file condiviso. `config`

  - **`ini`**provider: l'SDK cerca AWS le credenziali o le informazioni sul ruolo IAM nel file condiviso. `config`

- **`ecsCredentials`**provider - L'SDK cerca le variabili di ambiente `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` o `AWS_CONTAINER_CREDENTIALS_FULL_URI` che forniscono informazioni per acquisire credenziali temporanee.
- **`instanceProfile`**provider - L'SDK utilizza il servizio EC2 Instance Metadata per ottenere il ruolo IAM specificato nel profilo dell'istanza. Utilizzando le informazioni sul ruolo, l'SDK acquisisce credenziali temporanee.

> **Note**
>
> Il risultato del provider di default viene sottoposto automaticamente a memorizzazione.

È possibile esaminare il codice della catena nel codice sorgente. GitHub

Provider integrati nell'SDK

L'SDK offre diversi provider integrati che è possibile utilizzare singolarmente o combinare in una catena di fornitori di credenziali personalizzata.

Quando specificate un provider di credenziali durante la creazione del client di servizio, l'SDK tenta di caricare le credenziali utilizzando solo il provider di credenziali specificato. Non utilizza la catena di provider di credenziali predefinita. Se sai che desideri che un client di servizio utilizzi il `instanceProfile` provider, puoi cortocircuitare la catena predefinita specificando il `instanceProfile` provider nel costruttore del client di servizio:

```
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**
>
> I provider di credenziali vengono chiamati ogni volta che viene eseguita un'operazione API. Se il caricamento delle credenziali è un'operazione impegnativa (ad esempio perché avviene da disco o da una risorsa di rete) o se le credenziali non sono memorizzate nella cache dal provider, valutare se eseguire il wrapping del provider di credenziali in una funzione `Aws\Credentials\CredentialProvider::memoize`. Il provider di credenziali di default utilizzato dall'SDK viene sottoposto automaticamente a memorizzazione.

Argomenti

- assumeRolefornitore
- ssofornitore
- defaultProviderfornitore
- ecsCredentialsfornitore
- envfornitore
- assumeRoleWithWebIdentityCredentialProviderfornitore
- inifornitore
- processfornitore
- instanceProfilefornitore

`assumeRole`fornitore

Se utilizzi `Aws\Credentials\AssumeRoleCredentialProvider` per creare le credenziali tramite l'assunzione di un ruolo, devi trasmettere l'informazione `'client'` con un oggetto `StsClient` e dettagli `'assume_role_params'`, come illustrato.

> **Note**
>
> Per evitare di recuperare inutilmente AWS STS le credenziali su ogni operazione API, puoi utilizzare la `memoize` funzione per gestire l'aggiornamento automatico delle credenziali quando scadono. Di seguito è riportato un esempio di codice.

```
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
]);
```

Per ulteriori informazioni in merito, vedere. `'assume_role_params'` AssumeRole

`sso`fornitore

`Aws\Credentials\CredentialProvider::sso`è il provider di credenziali Single Sign-On. Questo provider è noto anche come fornitore di credenziali. AWS IAM Identity Center

```
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
]);
```

Se utilizzi un profilo denominato, sostituisci il nome del tuo profilo con '`default`' nell'esempio precedente. Per ulteriori informazioni sulla configurazione di profili denominati, consulta `configShared and credentials files` nella AWS SDKs and Tools Reference Guide. In alternativa, è possibile utilizzare la variabile di `AWS_PROFILE`ambiente per specificare le impostazioni del profilo da utilizzare.

Per saperne di più su come funziona il provider IAM Identity Center, consulta Understand IAM Identity Center authentication nella AWS SDKs and Tools Reference Guide.

`defaultProvider`fornitore

`Aws\Credentials\CredentialProvider::defaultProvider`è il provider di credenziali predefinito ed è anche chiamato catena di provider di credenziali predefinita. che viene utilizzato in caso di omissione dell'opzione `credentials` durante la creazione di un client. Ad esempio, se crei un S3Client come mostrato nel seguente frammento di codice, l'SDK utilizza il provider predefinito:

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

È inoltre possibile utilizzare DefaultProvider nel codice se si desidera fornire parametri a fornitori di credenziali specifici nella catena. Ad esempio, l'esempio seguente fornisce impostazioni personalizzate per il timeout della connessione e il ritentativo se viene utilizzata la funzione provider. `ecsCredentials`

```
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
]);
```

`ecsCredentials`fornitore

`Aws\Credentials\CredentialProvider::ecsCredentials` tenta di caricare le credenziali tramite una richiesta `GET`, il cui URI è specificato dalla variabile di ambiente `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` nel container.

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

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

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

`env`fornitore

L'utilizzo di variabili di ambiente per contenere le credenziali impedisce di condividere accidentalmente la chiave di accesso AWS segreta. Ti consigliamo di non aggiungere mai le chiavi di AWS accesso direttamente al client in nessun file di produzione.

Per autenticarsi su Amazon Web Services, l'SDK verifica innanzitutto le credenziali nelle variabili di ambiente. L'SDK utilizza la funzione `getenv()` per individuare le variabili di ambiente `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`e `AWS_SESSION_TOKEN`. Queste credenziali

vengono definite credenziali di ambiente. Per istruzioni su come ottenere questi valori, consulta Autenticazione con credenziali a breve termine nella and Tools Reference Guide.AWS SDKs

Se la tua applicazione è ospitata su AWS Elastic Beanstalk, puoi impostare le variabili `AWS_ACCESS_KEY_IDAWS_SECRET_KEY`, e di `AWS_SESSION_TOKEN` ambiente tramite la AWS Elastic Beanstalk console in modo che l'SDK possa utilizzare tali credenziali automaticamente.

Per ulteriori informazioni su come impostare le variabili di ambiente, consulta Supporto per le variabili di ambiente nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti. Inoltre, per un elenco di tutte le variabili di ambiente supportate dalla maggior parte AWS SDKs, consultate Elenco delle variabili di ambiente.

È inoltre possibile impostare le variabili di ambiente nella riga di comando, come illustrato di seguito.

Linux

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
   # The access key for your Account AWS.
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your Account AWS.
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your Account 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 Account AWS.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   # The secret access key for your Account AWS.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
   # The temporary session key for your Account 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` tenta di caricare le credenziali dalle variabili di ambiente.

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

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

`assumeRoleWithWebIdentityCredentialProvider`fornitore

`Aws\Credentials\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider` tenta di caricare le credenziali tramite l'assunzione di un ruolo. Se le variabili di ambiente `AWS_ROLE_ARN` e `AWS_WEB_IDENTITY_TOKEN_FILE` sono presenti, il provider cercherà di assumere il ruolo specificato in `AWS_ROLE_ARN` utilizzando il token su disco nel percorso completo specificato in `AWS_WEB_IDENTITY_TOKEN_FILE`. Se vengono utilizzate variabili di ambiente, il provider cercherà di impostare la sessione dalla variabile di ambiente `AWS_ROLE_SESSION_NAME`.

Se le variabili di ambiente non sono impostate, il provider utilizzerà il profilo predefinito oppure quello impostato come `AWS_PROFILE`. Il provider legge i profili da `~/.aws/credentials` e `~/.aws/config` per impostazione predefinita e può leggere i profili specificati nell'opzione di configurazione `filename`. Il provider assume il ruolo specificato in `role_arn` del profilo, leggendo un token dal percorso completo impostato in `web_identity_token_file`. `role_session_name` verrà utilizzato se impostato nel profilo.

Il provider viene chiamato come parte della catena predefinita e può essere chiamato direttamente.

```
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
```

```
]);
```

Per impostazione predefinita, questo provider di credenziali erediterà la regione configurata che verrà utilizzata da per assumere il ruolo. StsClient Facoltativamente, StsClient può essere fornito un file completo. Le credenziali devono essere impostate come `false` su quelle fornite. 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
]);
```

`ini`fornitore

`Aws\Credentials\CredentialProvider::ini`tenta di caricare le credenziali dai `credentials` file condivisi `config` e. Per impostazione predefinita, l'SDK tenta di caricare il profilo «predefinito» dal AWS `credentials` file condiviso che si trova in. `~/.aws/credentials` Se l'SDK trova la variabile di `AWS_SDK_LOAD_NONDEFAULT_CONFIG` ambiente, verifica anche la presenza di un profilo «predefinito» nel AWS `config` file condiviso che si trova in. `~/.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
]);
```

Puoi utilizzare un profilo o un percorso del file .ini personalizzato fornendo argomenti alla funzione che crea il provider.

```
$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`fornitore

`Aws\Credentials\CredentialProvider::process`tenta di caricare le credenziali eseguendo il `credential_process` valore specificato in un profilo in un `credentials`file condiviso `config` e.

Per impostazione predefinita, l'SDK tenta di caricare prima il profilo «predefinito» dal AWS `credentials` file condiviso che si trova in. `~/.aws/credentials` Se il profilo «predefinito» non viene trovato nel `credentials` file condiviso, l'SDK cerca il profilo predefinito nel `config` file condiviso. Di seguito è riportato un esempio di configurazione per il `credentials` file condiviso.

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

L'SDK chiamerà il `credential_process` comando esattamente come indicato utilizzando la `shell_exec` funzione di PHP e quindi leggerà i dati JSON da stdout. `credential_process`Devono scrivere le credenziali su stdout nel seguente formato:

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

`SessionToken` e `Expiration` sono facoltativi. Se presenti, le credenziali verranno considerate come temporanee.

```
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
]);
```

Puoi utilizzare un profilo o un percorso del file .ini personalizzato fornendo argomenti alla funzione che crea il provider.

```
$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`fornitore

`Aws\Credentials\CredentialProvider::instanceProfile`tenta di caricare le credenziali per un ruolo IAM specificato in un profilo di EC2 istanza Amazon.

```
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
]);
```

Per impostazione predefinita, il provider effettua un nuovo tentativo di recupero delle credenziali per un massimo di tre volte. Il numero di tentativi può essere impostato con l'`retries`opzione e disabilitato completamente impostando l'opzione su `0` come mostrato nel codice seguente.

```
use Aws\Credentials\CredentialProvider;

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

Se la variabile di ambiente `AWS_METADATA_SERVICE_NUM_ATTEMPTS` è disponibile, il suo valore ha la precedenza sull'opzione 'retries' mostrata in precedenza.

> **Note**
>
> Puoi disabilitare questo tentativo di caricamento dai profili delle EC2 istanze Amazon impostando la variabile di `AWS_EC2_METADATA_DISABLED` ambiente su`true`.

Concatenamento dei fornitori

I provider di credenziali possono essere concatenati utilizzando la funzione `Aws\Credentials\CredentialProvider::chain()`, che accetta un numero di argomenti variadic, ciascuno dei quali è una funzione del provider di credenziali. La funzione restituisce quindi una nuova funzione che è la composizione delle funzioni fornite, in modo tale che vengano richiamate una dopo l'altra finché uno dei provider non restituisce una promessa che viene soddisfatta correttamente.

Il `defaultProvider` utilizza questa composizione per verificare la presenza di più provider prima di generare errori. Nell'origine del `defaultProvider` è illustrato l'uso della funzione `chain`.

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

Creazione di un provider personalizzato

I provider di credenziali sono semplicemente funzioni che, se richiamate, restituiscono una promessa (`GuzzleHttp\Promise\PromiseInterface`) che viene soddisfatta con un oggetto `Aws\Credentials\CredentialsInterface` o rifiutata con un'eccezione `Aws\Exception\CredentialsException`.

Una delle best practice per la creazione di provider prevede la creazione di una funzione che viene richiamata per creare il provider di credenziali vero e proprio. Ad esempio ecco l'origine del provider `env` (leggermente modificata a scopo esemplificativo). Ricorda che si tratta di una funzione che restituisce la funzione di provider vera e propria. In questo modo puoi comporre facilmente i provider di credenziali e passarli come valori.

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

Memorizzazione delle credenziali

Talvolta è necessario creare un provider di credenziali in grado di ricordare il valore restituito in precedenza. Può essere utile per migliorare le prestazioni quando il caricamento di credenziali è un'operazione impegnativa o quando si usa la classe `Aws\Sdk` per condividere un provider di credenziali su più client. Puoi aggiungere la memorizzazione a un provider di credenziali eseguendo il wrapping della relativa funzione in una funzione `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());
```

Quando le credenziali sottoposte a memorizzazione scadono, il wrapper della funzione memoize richiama il provider nel wrapping nel tentativo di aggiornare le credenziali.

Assumi un ruolo IAM

Utilizzo dei ruoli IAM per le credenziali delle variabili di EC2 istanza Amazon

Se esegui la tua applicazione su un' EC2 istanza Amazon, il modo preferito per fornire le credenziali a cui effettuare chiamate AWS è utilizzare un ruolo IAM per ottenere credenziali di sicurezza temporanee.

Quando utilizzi i ruoli IAM, non devi preoccuparti della gestione delle credenziali della tua applicazione. Consentono a un'istanza di «assumere» un ruolo recuperando credenziali temporanee dal server di metadati dell' EC2 istanza Amazon.

Le credenziali temporanee, spesso chiamate credenziali del profilo di istanza, consentono l'accesso alle azioni e alle risorse consentite dalla politica del ruolo. Amazon EC2 gestisce tutte le fasi di autenticazione sicura delle istanze sul servizio IAM per assumere il ruolo e di aggiornare periodicamente le credenziali del ruolo recuperate. In questo modo, l'applicazione resta protetta quasi senza necessità del tuo intervento. Per un elenco dei servizi che accettano credenziali di sicurezza temporanee, consulta i servizi che funzionano con IAM nella Guida per l'AWS utente IAM.

> **Note**
>
> Per evitare di coinvolgere ogni volta il servizio dei metadati, è possibile trasmettere al costruttore di client un'istanza di `Aws\CacheInterface` come opzione `'credentials'`. In questo modo l'SDK utilizza le credenziali del profilo di istanza memorizzate nella cache. Per i dettagli, consulta Configurazione per la AWS SDK per PHP versione 3.

Per ulteriori informazioni sullo sviluppo di EC2 applicazioni Amazon utilizzando la SDKs, consulta Using IAM roles for Amazon EC2 instances nella AWS SDKs and Tools Reference Guide.

Crea e assegna un ruolo IAM a un'istanza Amazon EC2

1. Crea un client IAM.

Importazioni

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

use Aws\Iam\IamClient;
```

Codice di esempio

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

2. Crea un ruolo IAM con le autorizzazioni per le azioni e le risorse che utilizzerai.

Codice di esempio

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

3. Crea un profilo di istanza IAM e archivia l'Amazon Resource Name (ARN) dal risultato.

> **Note**
>
> Se utilizzi la console IAM anziché AWS SDK per PHP, la console crea automaticamente un profilo di istanza e gli assegna lo stesso nome del ruolo a cui corrisponde.
> Codice di esempio
>
> ```
> $IPN = 'InstanceProfileName';
>
> $result = $client->createInstanceProfile([
>     'InstanceProfileName' => $IPN ,
> ]);
>
> $ARN = $result['Arn'];
> $InstanceID =  $result['InstanceProfileId'];
> ```

4. Crea un EC2 client Amazon.

   Importazioni

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

   use Aws\Ec2\Ec2Client;
   ```

   Codice di esempio

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

5. Aggiungi il profilo dell'istanza a un' EC2 istanza Amazon in esecuzione o interrotta. Usa il nome del profilo dell'istanza del tuo ruolo IAM.

   Codice di esempio

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

Per ulteriori informazioni, consulta IAM Roles for Amazon EC2 nella Amazon EC2 User Guide.

Utilizzo dei ruoli IAM per le attività di Amazon ECS

Un'attività in Amazon Elastic Container Service (Amazon ECS) può assumere un ruolo IAM per AWS effettuare chiamate API. Si tratta di una strategia per la gestione delle credenziali da utilizzare per le applicazioni, simile a come i profili di EC2 istanza Amazon forniscono le credenziali alle istanze Amazon EC2 .

Invece di creare e distribuire AWS credenziali a lungo termine ai contenitori o utilizzare il ruolo dell' EC2 istanza Amazon, puoi associare un ruolo IAM che utilizza credenziali temporanee a una definizione di attività ECS o a un'operazione API. `RunTask`

Per ulteriori informazioni sull'utilizzo dei ruoli IAM che le attività del contenitore possono assumere, consulta l'argomento relativo al ruolo Task IAM nella Amazon ECS Developer Guide. Per esempi di utilizzo del ruolo IAM dell'attività sotto forma di una `taskRoleArn` nelle definizioni delle attività, consulta Definizioni di attività di esempio anche nella Amazon ECS Developer Guide.

Assumere un ruolo IAM in un altro Account AWS

Quando lavori in un Account AWS (Account A) e desideri assumere un ruolo in un altro account (Account B), devi prima creare un ruolo IAM nell'Account B. Questo ruolo consente alle entità del tuo account (Account A) di eseguire azioni specifiche nell'Account B. Per ulteriori informazioni sull'accesso tra account, consulta Tutorial: Delegate l'accesso tra AWS account tramite ruoli IAM.

Dopo aver creato un ruolo nell'Account B, annota il relativo ARN. Utilizzerai questo ARN quando assumerai il ruolo dall'Account A. Assumi il ruolo utilizzando AWS le credenziali associate alla tua entità nell'Account A.

Crea un AWS STS cliente con le credenziali per il tuo. Account AWS Nel seguente esempio, abbiamo utilizzato un profilo con credenziali, ma puoi utilizzare qualsiasi metodo. Con il client AWS STS appena creato, invoca assume-role e fornisci un valore di sessionName personalizzato. Recupera le nuove credenziali provvisorie dal risultato. Per impostazione predefinita, le credenziali durano un'ora.

Codice di esempio

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

Per ulteriori informazioni, consulta Using IAM Roles o AssumeRolenell' AWS SDK per PHP API Reference.

Utilizzo di un ruolo IAM con identità web

Web Identity Federation consente ai clienti di utilizzare provider di identità di terze parti per l'autenticazione durante l'accesso alle AWS risorse. Prima di assumere un ruolo con un'identità web, devi creare un ruolo IAM e configurare un provider di identità web (IdP). Per ulteriori informazioni, consultare Creazione di un ruolo per la federazione di identità web o di OpenID Connect (Console).

Dopo aver creato un provider di identità e creato un ruolo per la tua identità web, utilizza un AWS STS client per autenticare un utente. Fornisci l' webIdentityToken e ProviderId per la tua identità e il Role ARN per il ruolo IAM con le autorizzazioni per l'utente.

Codice di esempio

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

Per ulteriori informazioni, consulta AssumeRoleWithWebIdentity—Federation Through a Web-based Identity Provider o AssumeRoleWithWebIdentitynell'API Reference. AWS SDK per PHP

Assumi un ruolo con profilo

Definisci i profili in **`~/.aws/credentials`**

È possibile configurare l'utilizzo AWS SDK per PHP di un ruolo IAM definendo un profilo in`~/.aws/credentials`.

Crea un nuovo profilo con l'`role_arn`impostazione per il ruolo che vuoi assumere. Includi anche l'`source_profile`impostazione per un altro profilo con credenziali autorizzate ad assumere il ruolo IAM. Per maggiori dettagli su queste impostazioni di configurazione, consulta Assume le credenziali del ruolo nella AWS SDKs and Tools Reference Guide.

Ad esempio, di seguito`~/.aws/credentials`, il `project1` profilo imposta `role_arn` e specifica il `default` profilo come fonte per le credenziali per verificare che l'entità ad esse associata possa assumere il ruolo.

```
[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
```

Se si imposta la variabile di `AWS_PROFILE` ambiente o si utilizza il `profile` parametro quando si crea un'istanza di un client di servizio, `project1` viene assunto il ruolo specificato in, utilizzando il `default` profilo come credenziali di origine.

Il frammento seguente mostra l'uso del parametro in un costruttore. `profile` `S3Client` `S3Client`Avranno le autorizzazioni associate al ruolo associato al profilo. `project1`

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

Definire i profili in **`~/.aws/config`**

Il `~/.aws/config` file può contenere anche i profili che si desidera vengano assunti. Se imposti la variabile di ambienteAWS_SDK_LOAD_NONDEFAULT_CONFIG, l'SDK for PHP carica i profili `config` dal file. Quando AWS_SDK_LOAD_NONDEFAULT_CONFIG è impostato, l'SDK carica i profili da entrambi e. `~/.aws/config` `~/.aws/credentials` I profili di `~/.aws/credentials` vengono caricati per ultimi e hanno la precedenza su un profilo `~/.aws/config` con lo stesso nome. I profili della posizione possono servire come `source_profile` o il profilo da assumere.

L'esempio seguente utilizza il `project1` profilo definito nel `config` file e il `default` profilo nel `credentials` file. AWS_SDK_LOAD_NONDEFAULT_CONFIGÈ inoltre impostato.

```
# 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
```

Quando viene eseguito il `S3Client` costruttore, come mostrato nel frammento seguente, il ruolo definito nel `project1` profilo verrà assunto utilizzando le credenziali associate al profilo. `default`

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

```
]);
```

Usa credenziali temporanee da AWS STS

AWS Security Token Service (AWS STS) consente di richiedere privilegi limitati, credenziali temporanee per gli utenti IAM o per gli utenti autenticati tramite la federazione delle identità. Per una comprensione più approfondita, consulta Temporary Security Credentials nella IAM User Guide. Puoi utilizzare credenziali di sicurezza temporanee per accedere alla maggior parte dei AWS servizi. Per un elenco dei servizi che accettano credenziali di sicurezza temporanee, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Un caso d'uso comune delle credenziali temporanee consiste nel concedere alle applicazioni mobili o lato client l'accesso alle AWS risorse autenticando gli utenti tramite provider di identità di terze parti (vedi Web Identity Federation).

Ottenere credenziali temporanee

AWS STS dispone di diverse operazioni che restituiscono credenziali temporanee, ma l'`GetSessionToken`operazione è la più semplice da dimostrare. Il seguente frammento recupera le credenziali temporanee chiamando il `getSessionToken` metodo del client STS di PHP SDK.

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

$stsClient = $sdk->createSts();

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

Il risultato `GetSessionToken` e le altre AWS STS operazioni contengono sempre un valore. `'Credentials'` Se si stampa `$result` (ad esempio utilizzando`print_r($result)`), il risultato è simile al seguente.

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

Fornire credenziali temporanee a AWS SDK per PHP

È possibile utilizzare credenziali temporanee con un altro AWS client creando un'istanza del client e trasmettendo i valori ricevuti direttamente da. 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']
    ]
]);
```

Puoi anche costruire un oggetto `Aws\Credentials\Credentials` e utilizzarlo al momento di creare un'istanza del client.

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

$result = $stsClient->getSessionToken();

$credentials = new Credentials(
    $result['Credentials']['AccessKeyId'],
    $result['Credentials']['SecretAccessKey'],
    $result['Credentials']['SessionToken']
);

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

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

Tuttavia, il modo migliore per fornire le credenziali temporanee è utilizzare il metodo helper `createCredentials()` incluso in `StsClient`, Questo metodo estrae i dati da un AWS STS risultato e crea l'`Credentials`oggetto automaticamente.

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

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

Per ulteriori informazioni sul motivo per cui potrebbe essere necessario utilizzare credenziali temporanee nell'applicazione o nel progetto, consulta Scenari per la concessione dell'accesso temporaneo nella documentazione. AWS STS

Crea clienti anonimi

In alcuni casi, è consigliabile creare un client che non sia associato a nessuna credenziale, perché consente di effettuare richieste anonime a un servizio.

Ad esempio, puoi configurare sia gli oggetti Amazon S3 che i CloudSearch domini Amazon per consentire l'accesso anonimo.

Per creare un client anonimo, imposta l'opzione `'credentials'` su `false`.

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

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

```
]);
```

Oggetti di comando nella AWS SDK per PHP versione 3

AWS SDK per PHP utilizza il modello di comando per incapsulare i parametri e il gestore che verranno utilizzati per trasferire una richiesta HTTP in un secondo momento.

Uso implicito dei comandi

Se si esaminano le classi di client, è possibile vedere che i metodi corrispondenti alle operazioni delle API in realtà non esistono. La loro implementazione avviene utilizzando il metodo magico `__call()`. Questi pseudo-metodi sono in realtà collegamenti che incapsulano l'utilizzo degli oggetti di comando di SDK.

In genere, non è necessario interagire direttamente con gli oggetti di comando. Quando si chiamano metodi come `Aws\S3\S3Client::putObject()`, in realtà SDK crea un `Aws\CommandInterface` in base ai parametri forniti, esegue il comando e restituisce un oggetto `Aws\ResultInterface` popolato (o genera un'eccezione in caso di errore). Un flusso simile si verifica quando viene chiamato un metodo `Async` di un client (ad esempio, `Aws\S3\S3Client::putObjectAsync()`): il client crea un comando in base ai parametri forniti, serializza una richiesta HTTP, avvia la richiesta e restituisce una promessa.

I seguenti esempi sono equivalenti da un punto di vista funzionale.

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

$params = [
    'Bucket' => 'foo',
    'Key'    => 'baz',
    'Body'   => 'bar'
];

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

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

```
$result = $s3Client->execute($command);
```

Parametri di comando

Tutti i comandi supportano alcuni parametri speciali che non fanno parte di un'API del servizio, ma che controllano invece il comportamento di SDK.

`@http`

Quando si utilizza questo parametro, è possibile ottimizzare il modo in cui il gestore HTTP sottostante esegue la richiesta. Le opzioni che possono essere incluse nel parametro `@http` sono le stesse che possono essere impostate quando si crea un'istanza del client con l'opzione client "http".

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

`@retries`

Come nel caso dell'opzione client "nuovi tentativi", `@retries` controlla il numero massimo di nuovi tentativi di un comando prima che venga considerato non riuscito. Impostalo su `0` per disabilitare i nuovi tentativi.

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

> **Note**
>
> Se i nuovi tentativi sono stati disabilitati per un client, non è possibile abilitarli in modo selettivo su singoli comandi passati al client in questione.

Creazione di oggetti di comando

È possibile creare un comando utilizzando il metodo `getCommand()` di un client. Non esegue o trasferisce immediatamente una richiesta HTTP, ma viene eseguito soltanto quando viene passato al metodo `execute()` del client. In questo modo, è possibile modificare l'oggetto di comando prima di eseguire il comando.

```
$command = $s3Client->getCommand('ListObjects');
$command['MaxKeys'] = 50;
$command['Prefix'] = 'foo/baz/';
$result = $s3Client->execute($command);

// You can also modify parameters
$command = $s3Client->getCommand('ListObjects', [
    'MaxKeys' => 50,
    'Prefix'  => 'foo/baz/',
]);
$command['MaxKeys'] = 100;
$result = $s3Client->execute($command);
```

Comando **`HandlerList`**

Quando un comando viene creato da un client, gli viene assegnato un clone dell'oggetto `Aws\HandlerList` del client. Al comando viene assegnato un clone dell'elenco dei gestori del client per consentire a un comando di utilizzare middleware e gestori personalizzati che non interessano altri comandi eseguiti dal client.

In questo modo, è possibile utilizzare un client HTTP diverso per ogni comando (ad esempio, `Aws\MockHandler`) e aggiungere un comportamento personalizzato per ogni comando tramite middleware. I seguenti esempi utilizzano un `MockHandler` per creare risultati fittizi anziché inviare richieste HTTP effettive.

```
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'
```

Oltre a modificare il gestore utilizzato dal comando, è anche possibile includere un middleware personalizzato nel comando. L'esempio seguente utilizza il middleware `tap`, che funge da osservatore nell'elenco dei gestori.

```
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` consente di eseguire comandi simultaneamente utilizzando un'iterazione che produce oggetti `Aws\CommandInterface`. `CommandPool` garantisce che un numero costante di comandi venga eseguito simultaneamente durante l'iterazione sui comandi nel pool (mentre i comandi vengono completati, ne vengono eseguiti altri per garantire dimensioni del pool costanti).

Di seguito è riportato un semplice esempio di invio di alcuni comandi utilizzando `CommandPool`.

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

// Create the client
$client = new S3Client([
    'region'  => 'us-standard',
```

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

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

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

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

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

Questo esempio per `CommandPool` è piuttosto elementare. L'esempio seguente è più complesso. Supponiamo che tu voglia caricare file su disco in un bucket Amazon S3. Per ottenere un elenco dei file del disco, è possibile utilizzare la funzione `DirectoryIterator` di PHP. Questa iterazione genera oggetti `SplFileInfo`. `CommandPool` accetta un'iterazione che genera oggetti `Aws\CommandInterface`, perciò occorre eseguire la mappatura sugli oggetti `SplFileInfo` per restituire oggetti `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"; });
```

Configurazione **`CommandPool`**

Il costruttore `Aws\CommandPool` accetta varie opzioni di configurazione.

concurrency (callable|int)

> Numero massimo di comandi da eseguire simultaneamente. È possibile fornire una funzione per ridimensionare il pool in modo dinamico. La funzione include l'attuale numero di richieste in sospeso e dovrebbe restituire un numero intero che rappresenta il nuovo limite delle dimensioni del pool.

before (callable)

> Funzione che consente di eseguire l'invocazione prima dell'invio di un comando. La funzione `before` accetta il comando e la chiave dell'iterazione del comando. È possibile modificare il comando secondo necessità nella funzione `before` prima di inviare il comando.

fulfilled (callable)

> Funzione che consente di effettuare l'invocazione quando una promessa viene soddisfatta. La funzione include l'oggetto risultato, l'ID dell'iterazione da cui proveniva il risultato e la promessa in forma aggregata che può essere risolta o respinta se occorre cortocircuitare il pool.

rejected (callable)

Funzione che consente di effettuare l'invocazione quando una promessa viene respinta. La funzione include un oggetto `Aws\Exception`, l'ID dell'iterazione da cui proveniva l'eccezione e la promessa in forma aggregata che può essere risolta o respinta se occorre cortocircuitare il pool.

Raccolta manuale dei rifiuti tra i comandi

Se pool di comandi di grandi dimensioni causano il raggiungimento del limite di memoria, la causa è da ricercarsi nei riferimenti ciclici generati dall'SDK che non sono ancora stati raccolti dal garbage collector PHP quando il limite di memoria è stato raggiunto. L'invocazione manuale dell'algoritmo di raccolta tra i comandi può consentire la raccolta dei cicli prima del raggiungimento di tale limite. L'esempio seguente crea un `CommandPool` che richiama l'algoritmo di raccolta utilizzando un callback prima dell'invio di ogni comando. L'invocazione di garbage collector comporta costi di prestazioni e l'uso ottimale dipende dal caso d'uso e dall'ambiente.

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

Promesse nella AWS SDK per PHP versione 3

L' AWS SDK per PHP utilizzo promette di consentire flussi di lavoro asincroni e questa asincronicità consente l'invio simultaneo di richieste HTTP. La specifica per le promesse utilizzata da SDK è Promises/A+.

Cos'è una promessa?

Una promessa rappresenta l'eventuale risultato di un'operazione asincrona. La modalità principale di interazione con una promessa è attraverso il metodo `then`. Questo metodo registra callback per ricevere l'eventuale valore di una promessa o il motivo per cui la promessa non può essere soddisfatta.

Si AWS SDK per PHP affida al pacchetto guzzlehttp/promise Composer per l'implementazione delle promesse. Le promesse Guzzle supportano flussi di lavoro bloccanti e non bloccanti e possono essere utilizzate con qualsiasi loop di eventi non bloccanti.

> **Note**
>
> Le richieste HTTP vengono inviate contemporaneamente AWS SDK per PHP utilizzando un singolo thread, in cui vengono utilizzate chiamate non bloccanti per trasferire una o più richieste HTTP reagendo ai cambiamenti di stato (ad esempio, adempiendo o rifiutando le promesse).

Promesse in SDK

Le promesse vengono utilizzate in tutto il kit SDK. Ad esempio, le promesse vengono utilizzate nella maggior parte delle astrazioni di alto livello fornite da SDK: impaginatori, waiter, pool di comandi, caricamenti in più parti, trasferimenti tra directory S3 e bucket e così via.

Tutti i client che SDK fornisce restituiscono promesse quando si invoca uno qualsiasi dei metodi con suffisso `Async`. Ad esempio, il codice seguente mostra come creare una promessa per ottenere i risultati di un'operazione 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']);
```

Nota: è possibile è effettuare una chiamata a `describeTable` o a `describeTableAsync`. Questi metodi sono metodi magici `__call` su un client supportati dal modello di API e dal numero `version` associati al client. Chiamando metodi quali `describeTable` senza il suffisso `Async`, il client presenterà un blocco mentre invia una richiesta HTTP e restituirà un oggetto `Aws\ResultInterface` o genererà una `Aws\Exception\AwsException`. Aggiungendo al nome dell'operazione il suffisso `Async` (ad esempio, `describeTableAsync`), il client creerà una promessa che verrà soddisfatta con un oggetto `Aws\ResultInterface` o rifiutata con una `Aws\Exception\AwsException`.

> **Important**
>
> Quando la promessa viene restituita, il risultato potrebbe essere già arrivato (ad esempio, quando si utilizza un gestore fittizio), oppure la richiesta HTTP potrebbe non essere stata avviata.

È possibile registrare un callback con la promessa utilizzando il metodo `then`. Questo metodo accetta due callback, `$onFulfilled` e `$onRejected`, entrambi opzionali. Il callback `$onFulfilled` viene invocato se la promessa viene soddisfatta, mentre il callback `$onRejected` viene invocato se la promessa viene respinta (ovvero ha esito negativo).

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

Esecuzione simultanea di comandi

È possibile comporre più promesse insieme in modo che vengano eseguite simultaneamente. A tal fine, è possibile integrare SDK con un loop di eventi non bloccante, oppure creare più promesse e attenderne il completamento simultaneo.

```
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**
>
> CommandPoolFornisce un meccanismo più potente per l'esecuzione simultanea di più operazioni API.

Concatenare le promesse

Uno degli aspetti migliori delle promesse è il fatto che sono componibili, per cui consentono di creare pipeline di trasformazione.. Le promesse vengono composte concatenando callback `then` con callback `then` successivi. Il valore restituito da un metodo `then` è una promessa che viene soddisfatta o respinta in base al risultato dei callback forniti.

```
$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**
>
> Il valore restituito dal callback di una promessa è l'argomento `$value` che viene fornito a promesse in downstream. Se si desidera fornire un valore per eseguire il downstream di catene di promesse, è necessario restituire un valore nella funzione di callback.

Inoltro del rifiuto

È possibile registrare un callback da invocare quando una promessa viene respinta. Se viene generata un'eccezione in un callback, la promessa viene respinta con quell'eccezione e le promesse successive nella catena vengono rifiutate con quell'eccezione. Se si restituisce correttamente un valore da un callback `$onRejected`, le promesse successive nella catena vengono soddisfatte con il valore restituito dal callback `$onRejected`.

Aspettando le promesse

È possibile forzare il completamento sincrono di una promessa utilizzando il metodo `wait` della promessa.

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

Se si presenta un'eccezione durante l'invocazione della funzione `wait` di una promessa, la promessa viene respinta con l'eccezione e l'eccezione viene sollevata.

```
use Aws\Exception\AwsException;

$promise = $client->listTablesAsync();

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

```
    // Handle the error
}
```

La chiamata della funzione `wait` su una promessa che è stata soddisfatta non attiva la funzione di attesa. Restituisce semplicemente il valore distribuito in precedenza.

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

La chiamata della funzione `wait` su una promessa che è stata respinta genera un'eccezione. Se il motivo del rifiuto è un'istanza di `\Exception`, il motivo viene generato. Altrimenti, viene generato `GuzzleHttp\Promise\RejectionException` e il motivo può essere ottenuto chiamando il metodo `getReason` dell'eccezione.

> **Note**
>
> Le chiamate operative API in AWS SDK per PHP vengono rifiutate con le sottoclassi della `Aws\Exception\AwsException` classe. Tuttavia, è possibile che il motivo distribuito a un metodo `then` sia diverso per via dell'aggiunta di un middleware personalizzato che modifica il motivo del rifiuto.

Annullamento delle promesse

Le promesse possono essere annullate utilizzando il metodo `cancel()` di una promessa. Se una promessa è stata già risolta , chiamare `cancel()` non avrà alcun effetto. L'annullamento di una promessa determina l'annullamento anche delle promesse in attesa di distribuzione dalla promessa. Una promessa annullata viene respinta con `GuzzleHttp\Promise\RejectionException`.

Combinare promesse

È possibile combinare promesse in forma aggregata per creare flussi di lavoro più sofisticati. Il pacchetto `guzzlehttp/promise` contiene varie funzioni che è possibile utilizzare per combinare promesse.

Puoi trovare la documentazione API per tutte le funzioni di raccolta delle promesse in namespace-GuzzleHttp .Promise.

each e each_limit

Utilizza la coda di `Aws\CommandInterface` comandi CommandPoolwhen you have a task da eseguire contemporaneamente a un pool di dimensioni fisse (i comandi possono essere in memoria o generati da un iteratore pigro). `CommandPool` garantisce che un numero fisso di comandi venga inviato simultaneamente fino a esaurimento dell'iterazione fornita.

`CommandPool` funziona solo con comandi eseguiti dal client stesso. È possibile utilizzare la funzione `GuzzleHttp\Promise\each_limit` per eseguire l'invio di comandi di diversi client simultaneamente utilizzando un pool di dimensioni fisse.

```
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 di promessa

Una delle caratteristiche più potenti della libreria di promesse Guzzle è il fatto che consente di utilizzare co-routine delle promesse che rendono la scrittura di flussi di lavoro asincroni più simile alla scrittura di tradizionali flussi di lavoro sincroni. E, in effetti, AWS SDK per PHP utilizza co-routine delle promesse nella maggior parte delle astrazioni di alto livello.

Si immagini di creare diversi bucket e di caricare un file nel bucket quando quest'ultimo diventa disponibile, il tutto simultaneamente, in modo che l'operazione venga eseguita nel minor tempo possibile. È possibile eseguire questa operazione in modo semplice combinando più co-routine utilizzando la funzione delle promesse `all()`.

```
use GuzzleHttp\Promise;

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

// Create the following buckets
$buckets = ['foo', 'baz', 'bar'];
$promises = [];

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

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

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

Gestori e middleware nella versione 3 AWS SDK per PHP

Il meccanismo principale per estendere il AWS SDK per PHP è tramite gestori e middleware. Ogni classe di client SDK possiede un'istanza `Aws\HandlerList` accessibile tramite il metodo `getHandlerList()` di un client. Puoi recuperare un `HandlerList` del client e modificarlo per aggiungere o rimuovere il comportamento client.

Gestori

Un gestore è una funzione che esegue l'effettiva trasformazione di un comando e richiesta in un risultato. Un gestore in genere invia richieste HTTP. I gestori possono essere composti con middleware per potenziare il comportamento. Un gestore è una funzione che accetta un `Aws\CommandInterface` e un `Psr\Http\Message\RequestInterface` e restituisce una promessa che viene soddisfatta con un `Aws\ResultInterface` o respinta con un motivo `Aws\Exception\AwsException`.

Ecco un gestore che restituisce lo stesso risultato fittizio per ogni chiamata.

```
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);
};
```

Puoi quindi utilizzare questo gestore con un client SDK fornendo un'opzione `handler` nel costruttore di un client.

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

Puoi anche modificare il gestore di un client dopo che è stata costruito utilizzando il metodo `setHandler` di un `Aws\ClientInterface`.

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

> **Note**
>
> Per modificare il gestore di un client multiregionale dopo la sua creazione, usa il metodo di un. `useCustomHandler` `Aws\MultiRegionClient`
>
> ```
> $multiRegionClient->useCustomHandler($myHandler);
> ```

Gestore fittizio

Ti consigliamo di usare il `MockHandler` quando scrivi test che utilizzano l'SDK. Puoi usare `Aws\MockHandler` per restituire risultati fittizi o generare eccezioni fittizie. I risultati o le eccezioni vengono accodati e poi li rimuove in ordine FIFO MockHandler .

```
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

Il middleware è uno speciale tipo di funzioni di alto livello che migliorano il comportamento di trasferimento di un comando ed eseguono la delega a un gestore "successivo". Le funzioni middleware accettano `Aws\CommandInterface` e `Psr\Http\Message\RequestInterface` e restituiscono una promessa che viene soddisfatta con `Aws\ResultInterface` o respinta con un motivo `Aws\Exception\AwsException`.

Un middleware è una funzione di ordine più alto che modifica un comando, richiesta o risultato quando passa attraverso il middleware. Un middleware ha la forma seguente.

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

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

Un middleware riceve un comando per l'esecuzione e un oggetto di richiesta opzionale. Il middleware può scegliere di potenziare la richiesta e il comando o di lasciarli senza alcuna modifica. Un middleware invoca quindi l'handle successivo nella catena o può scegliere di cortocircuitare il gestore successivo e restituire una promessa. La promessa creata invocando il gestore successivo può

essere potenziata utilizzando il metodo `then` della promessa di modificare il risultato o l'errore prima di restituire la promessa dello stack di middleware.

HandlerList

L'SDK usa un `Aws\HandlerList` per gestire il middleware e i gestori utilizzati quando si esegue un comando. Ogni client SDK possiede un `HandlerListe` questo `HandlerList` viene clonato e aggiunto a ogni comando creato da un client. Puoi collegare un middleware e gestore predefinito da utilizzare per ogni comando creato da un client aggiungendo un middleware all'`HandlerList` del client. Puoi aggiungere e rimuovere middleware da comandi specifici modificando l'`HandlerList` di proprietà di un comando specifico.

Un `HandlerList` rappresenta uno stack di middleware che vengono utilizzato per avvolgere un gestore. Per aiutarti a gestire l'elenco di middleware e l'ordine in cui avvolgono un handler, l'`HandlerList` divide lo stack middleware in passaggi denominati che rappresentano parte del ciclo di vita di trasferimento di un comando:

1. `init` - Aggiungi parametri di default
2. `validate` - Convalida parametri obbligatori
3. `build` - Serializza una richiesta HTTP per l'invio
4. `sign` - Firma la richiesta HTTP serializzata
5. **<**gestore **>** (non una fase, ma esegue il trasferimento effettivo)

init

Questa fase del ciclo di vita rappresenta l'inizializzazione di un comando e una richiesta non è stata ancora serializzata. Questa fase viene in genere utilizzata per aggiungere parametri di default a un comando.

Puoi aggiungere un middleware alla fase `init` usando i metodi `appendInit` e `prependInit`, dove `appendInit` aggiunge il middleware alla fine dell'elenco `prepend` mentre `prependInit` aggiunge il middleware all'inizio dell'elenco `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');
```

validate

Questa fase del ciclo di vita viene utilizzata per convalidare i parametri di input di un comando.

Puoi aggiungere un middleware alla fase `validate` usando i metodi `appendValidate` e `prependValidate`, dove `appendValidate` aggiunge il middleware alla fine dell'elenco `validate` mentre `prependValidate` aggiunge il middleware all'inizio dell'elenco `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

Questa fase del ciclo di vita viene utilizzata per serializzare una richiesta HTTP per il comando in esecuzione. Gli eventi del ciclo di vita a valle riceveranno un comando e una richiesta PSR-7 HTTP.

Puoi aggiungere un middleware alla fase `build` usando i metodi `appendBuild` e `prependBuild`, dove `appendBuild` aggiunge il middleware alla fine dell'elenco `build` mentre `prependBuild` aggiunge il middleware all'inizio dell'elenco `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');
```

Firma

Questa fase del ciclo di vita viene in genere utilizzata per firmare le richieste HTTP prima che siano inviate tramite la rete. È consigliabile in genere non mutare mai una richiesta HTTP dopo averla firmata per evitare errori di firma.

Questa è l'ultima fase di `HandlerList` prima che la richiesta HTTP viene trasferita da un gestore.

Puoi aggiungere un middleware alla fase `sign` usando i metodi `appendSign` e `prependSign`, dove `appendSign` aggiunge il middleware alla fine dell'elenco `sign` mentre `prependSign` aggiunge il middleware all'inizio dell'elenco `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 disponibile

L'SDK fornisce diversi middleware che puoi utilizzare per potenziare il comportamento di un client o per osservare l'esecuzione di un comando.

mapCommand

Il middleware `Aws\Middleware::mapCommand` è utile quando devi modificare un comando prima che il comando venga serializzato come una richiesta HTTP. Ad esempio, `mapCommand` può essere utilizzato per eseguire la convalida o aggiungere parametri di default. La funzione `mapCommand` accetta un chiamabile che accetta un oggetto `Aws\CommandInterface` e restituisce un oggetto `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

Il middleware `Aws\Middleware::mapRequest` è utile quando devi modificare una richiesta dopo che è stata serializzata ma prima che è stata inviata. Ad esempio, questo può essere utilizzato per aggiungere intestazioni HTTP personalizzate a una richiesta. La funzione `mapRequest` accetta un chiamabile che accetta un argomento `Psr\Http\Message\RequestInterface` e restituisce un oggetto `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'
);
```

Quando si esegue il comando, viene inviata con l'intestazione personalizzata.

> **Important**
>
> Si noti che il middleware è stato aggiunto all'elenco dei gestori alla fine della fase `build`. Questo è per far sì che una richiesta sia stata creata prima della chiamata del middleware.

mapResult

Il middleware `Aws\Middleware::mapResult` è utile quando devi modificare il risultato dell'esecuzione di un comando. La funzione `mapResult` accetta un chiamabile che accetta un argomento `Aws\ResultInterface` e restituisce un oggetto `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;
    })
);
```

Ora quando il comando viene eseguito, il risultato restituito conterrà un attributo `foo`.

cronologia

Il middleware `history` è utile per verificare che l'SDK abbia eseguito i comandi previsti, inviato le richieste HTTP previste e ricevuto i risultati previsti. È essenzialmente un middleware che agisce in modo analogo alla cronologia di un 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));
```

Un container cronologico `Aws\History` memorizza 10 voci per impostazione predefinita prima di rimuovere le voci. Puoi personalizzare il numero di voci passando il numero di voci che persistono al costruttore.

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

Puoi controllare il container cronologico dopo l'esecuzione di richieste che passano il middleware di cronologia.

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

// The object is iterable, yielding each entry in the container
foreach ($history as $entry) {
    // You can access the command that was executed
    var_dump($entry['command']);
    // The request that was serialized and sent
    var_dump($entry['request']);
    // The result that was received (if successful)
    var_dump($entry['result']);
    // The exception that was received (if a failure occurred)
    var_dump($entry['exception']);
}

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

// You can get the last request that was serialized. This method will throw an
 exception
// if no requests have been serialized.
```

```
$request = $history->getLastRequest();

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

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

tap

Il middleware `tap` viene utilizzato come osservatore. Puoi usare questo middleware per richiamare le funzioni quando invii comandi attraverso la catena di middleware. La funzione `tap` accetta un chiamabile che accetta `Aws\CommandInterface` e un ulteriore `Psr\Http\Message\RequestInterface` che viene eseguito.

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

Creazione di gestori personalizzati

Un gestore è semplicemente una funzione che accetta un oggetto `Aws\CommandInterface` e un oggetto `Psr\Http\Message\RequestInterface` e restituisce un `GuzzleHttp\Promise\PromiseInterface` che viene soddisfatto con un `Aws\ResultInterface` o rifiutato con un `Aws\Exception\AwsException`.

Sebbene l'SDK disponga di diverse opzioni `@http`, un gestore deve sapere solo come utilizzare le seguenti opzioni:

- connect_timeout
- debug
- decode_content (opzionale)
- delay
- progress (opzionale)
- proxy
- sink
- synchronous (opzionale)
- stream (opzionale)
- timeout
- Verifica
- http_stats_receiver (opzionale) - Una funzione per invocare con un array associativo di statistiche di trasferimento HTTP se richiesto utilizzando il parametro di configurazione stats.

A meno che l'opzione non sia specificata come opzionale, un gestore DEVE essere in grado di gestire l'opzione oppure DEVE restituire una promessa rifiutata.

Oltre a gestire `@http` opzioni specifiche, un gestore DEVE aggiungere un'`User-Agent`intestazione che assuma il seguente formato, dove «3.X» può essere sostituito con `Aws\Sdk::VERSION` e «HandlerSpecificData/version...» deve essere sostituito con la stringa User-Agent specifica del gestore.

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

Stream nella AWS SDK per PHP versione 3

Come parte della sua integrazione con lo standard di messaggi HTTP PSR-7, AWS SDK per PHP utilizza StreamInterface internamente la PSR-7 come astrazione sui flussi PHP. Qualsiasi comando con un campo di input definito come blob, ad esempio il `Body` parametro di un PutObject comando S3::, può essere soddisfatto con una stringa, una risorsa di flusso PHP o un'istanza di. `Psr\Http\Message\StreamInterface`

> **Warning**
>
> L'SDK assume la proprietà di qualsiasi risorsa di flusso PHP non elaborata fornita come parametro di input a un comando. Il flusso viene consumato e chiuso per tuo conto.
> Se devi condividere un flusso tra un'operazione SDK e il tuo codice, avvolgilo in un'istanza di `GuzzleHttp\Psr7\Stream` prima di includerlo come parametro di comando. L'SDK consuma il flusso, pertanto il tuo codice deve tenere conto del movimento del cursore interno del flusso. I flussi Guzzle chiamano `fclose` sulla risorsa di flusso sottostante quando vengono distrutti dal Garbage Collector di PHP, perciò non è necessario chiudere il flusso.

Decoratori Stream

Guzzle fornisce diversi decoratori di flussi che puoi utilizzare per controllare come l'SDK e Guzzle interagiscono con le risorse di streaming fornite come parametro di input a un comando. Questi decoratori possono modificare il modo in cui i gestori sono in grado di leggere e cercare su un determinato flusso. Di seguito è riportato un elenco parziale; ulteriori informazioni possono essere trovate nel repository GuzzleHttpPsr 7.

AppendStream

GuzzleHttp\ Psr7\ AppendStream

Le letture provenienti da più flussi, una dopo l'altra.

```
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

Utilizzato per consentire la ricerca sui byte precedentemente letti su flusso non ricercabili. Questo può essere utile quando il trasferimento di un corpo entità non ricercabile ha esito negativo a causa della necessità di riavvolgere il flusso (ad esempio, dopo un reindirizzamento). I dati che vengono letti dal flusso remoto sono eseguiti nel buffer in un flusso PHP temporaneo, in modo che i byte letti in precedenza vengano prima salvati nella cache in memoria, poi su disco.

```
use GuzzleHttp\Psr7;

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

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

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

InflateStream

GuzzleHttp\ Psr7\ InflateStream

Utilizza il filtro zlib.inflate di PHP per decomprimere o comprimere il contenuto gzipped.

Questo decoratore di flussi salta i primi 10 byte del flusso per eliminare l'intestazione gzip, converte il flusso fornito in una risorsa di flusso PHP, quindi aggiunge il filtro zlib.inflate. Il flusso viene poi riconvertito in una risorsa di flusso Guzzle per essere utilizzato come flusso Guzzle.

LazyOpenStream

GuzzleHttp\ Psr7\ LazyOpenStream

Esegue lentamente operazioni di lettura e scrittura di un file che viene aperto solo dopo un'operazione di I/O sul flusso.

```
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

Utilizzato per leggere un sottoinsieme o una porzione di un oggetto di flusso esistente. Questo può essere utile per suddividere un file di grandi dimensioni in parti più piccole da inviare in blocchi (ad esempio, l'API Amazon S3 Multipart Upload).

```
use GuzzleHttp\Psr7;

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

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

NoSeekStream

GuzzleHttp\ Psr7\ NoSeekStream

Racchiude un flusso e non consente di cercare.

```
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

Fornisce un flusso di sola lettura che pompa dati da un chiamabile PHP.

Quando si richiama il callable fornito, PumpStream passa la quantità di dati richiesti da leggere al callable. Il chiamabile può scegliere di ignorare questo valore e restituire un numero minore o più byte di quanto richiesto. Tutti i dati aggiuntivi restituiti dal callable fornito vengono memorizzati internamente nel buffer fino a quando non vengono scaricati utilizzando la funzione read () di. PumpStream Il chiamabile fornito DEVE restituire false quando non ci sono più dati da leggere.

Implementazione di decoratori di flussi

Creare un decoratore di stream è molto semplice grazie a GuzzleHttp\ Psr7\. StreamDecoratorTrait Questa caratteristica offre metodi che implementano `Psr\Http\Message\StreamInterface` con il proxy a un flusso sottostante. Basta `use StreamDecoratorTrait` e implementare i metodi personalizzati.

Supponiamo, ad esempio, che desideri chiamare una determinata funzione ogni volta che l'ultimo byte viene letto da un flusso. Questo potrebbe essere implementato sostituendo il metodo `read()`.

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

class EofCallbackStream implements StreamInterface
{
    use StreamDecoratorTrait;

    private $callback;

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

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

        // Invoke the callback when EOF is hit
```

```
        if ($this->eof()) {
            call_user_func($this->callback);
        }

        return $result;
    }
}
```

Questo decoratore potrebbe essere aggiunto a qualsiasi flusso esistente e utilizzato in questo modo.

```
use GuzzleHttp\Psr7;

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

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

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

Paginatori nella versione 3 AWS SDK per PHP

Alcune operazioni AWS di servizio sono suddivise in pagine e rispondono con risultati troncati. Ad esempio, l'`ListObjects`operazione Amazon S3 restituisce solo fino a 1.000 oggetti alla volta. Operazioni come queste (in genere contraddistinte dal prefisso "list" o "describe") necessitano di richieste successive con parametri di token (o contrassegno) per recuperare l'intero set di risultati.

I paginatori sono una funzionalità di The AWS SDK per PHP che funge da astrazione rispetto a questo processo per semplificare l'utilizzo di paginated da parte degli sviluppatori. APIs Un impaginatore è essenzialmente un'iterazione dei risultati, creati mediante il metodo `getPaginator()` del client. Quando richiami `getPaginator()`, devi fornire il nome e gli argomenti dell'operazione (come avviene per l'esecuzione di un'operazione). È possibile eseguire iterazioni sull'oggetto di un impaginatore utilizzando `foreach` per ottenere singoli oggetti `Aws\Result`.

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

```
]);

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

Oggetti Paginator

L'oggetto restituito dal metodo `getPaginator()` è un'istanza della classe `Aws\ResultPaginator`, che implementa l'interfaccia nativa PHP `iterator`, motivo per cui funziona con `foreach`. Può anche essere usato con funzioni di iterazione, come `iterator_to_array`, e si integra perfettamente con le iterazioni SPL come l'oggetto `LimitIterator`.

Gli oggetti dell'impaginatore contengono solo una "pagina" di risultati alla volta e vengono eseguiti lentamente: ciò significa che effettuano solo il numero di richieste necessarie a ottenere la pagina corrente dei risultati. Ad esempio, l'`ListObjects`operazione Amazon S3 restituisce solo fino a 1.000 oggetti alla volta, quindi se il bucket contiene circa 10.000 oggetti, l'impaginatore dovrebbe eseguire un totale di 10 richieste. Quando effettui l'iterazione dei risultati, la prima richiesta viene eseguita all'avvio dell'iterazione, la seconda alla seconda iterazione del loop e così via.

Enumerazione dei dati dai risultati

Gli oggetti dell'impaginatore dispongono di un metodo denominato `search()`, che consente di creare iterazioni per i dati all'interno di un set di risultati. Quando chiami`search()`, fornisci un'JMESPath espressione per specificare quali dati estrarre. La chiamata di `search()` restituisce un'iterazione che consente di ottenere i risultati dell'espressione in ogni pagina di risultati. La valutazione avviene in modo pigro durante lo scorrimento nell'iterazione restituita.

L'esempio che segue è equivalente al codice di esempio precedente, ma con l'utilizzo del metodo `ResultPaginator::search()` per brevità

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

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

JMESPath le espressioni consentono di eseguire operazioni piuttosto complesse. Se ad esempio vuoi stampare tutte le chiavi e i prefissi comuni dell'oggetto (ad esempio eseguire `ls` di un bucket), puoi procedere come segue.

```
// 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";
}
```

Impaginazione asincrona

Puoi eseguire iterazioni dei risultati di un impaginatore in modo asincrono fornendo un callback per il metodo `each()` di un `Aws\ResultPaginator`. Il callback viene richiamato per ogni valore ottenuto dall'impaginatore.

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

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

> **Note**
>
> L'utilizzo del metodo `each()` consente di impaginare i risultati di un'operazione API mentre vengono contestualmente inviate altre richieste in modo asincrono.

Un valore non nullo restituito dal callback verrà prodotto dalla coroutine sottostante basata sulla promessa: ciò significa dal callback possono essere restituite promesse che devono essere risolte prima di continuare l'iterazione sui restanti elementi, essenzialmente unendo altre promesse all'iterazione. L'ultimo valore non nullo restituito dal callback è il risultato che soddisfa la promessa

per qualsiasi promessa di tipo downstream. Se l'ultimo valore restituito è una promessa, la risoluzione di quest'ultima è il risultato che comporta il soddisfacimento o il rifiuto delle promesse di downstream.

```
// 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();
```

Camerieri nella versione 3 AWS SDK per PHP

I waiter aiutano a semplificare il lavoro con sistemi con coerenza finale fornendo un modo astratto per attendere che una risorsa entri in uno stato specifico effettuando il polling della risorsa. È possibile trovare un elenco dei camerieri supportati da un cliente visualizzando la documentazione API per una singola versione di un client di servizio. Per navigare lì, vai alla pagina del cliente nella documentazione dell'API e vai al numero di versione specifico (rappresentato da una data) e scorri verso il basso fino alla sezione «Camerieri». Questo link ti porterà alla sezione camerieri di S3.

Nell'esempio seguente, il client Amazon S3 viene utilizzato per creare un bucket. Quindi il waiter viene utilizzato per attendere fino all'esistenza del bucket.

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

Se il waiter deve eseguire il polling del bucket troppe volte, genera un'eccezione `\RuntimeException`.

Configurazione del cameriere

I waiter sono guidati da un array associativo di opzioni di configurazione. Tutte le opzioni utilizzate da un determinato waiter dispongono di valori predefiniti, ma essi possono essere sovrascritti per supportare diverse strategie di attesa.

È possibile modificare le opzioni di configurazione del waiter trasferendo un array associativo di opzioni `@waiter` all'argomento `$args` di un metodo di client `waitUntil()` e `getWaiter()`.

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

delay (int)

Numero di secondi di ritardo tra i tentativi di polling. Ogni waiter ha un valore di configurazione predefinito `delay` ma potrebbe non essere necessario modificare questa impostazione per casi d'uso specifici.

maxAttempts (int)

Il numero massimo di tentativi di polling da emettere prima che il waiter abbia esito negativo. Questa opzione garantisce che non si debba attendere una risorsa a tempo indeterminato. Ogni waiter ha un valore di configurazione predefinito `maxAttempts` ma potrebbe non essere necessario modificare questa impostazione per casi d'uso specifici.

initDelay (int)

Quantità di tempo di attesa in secondi prima del primo tentativo di polling. Questo può essere utile quando si attende una risorsa che richiederà qualche minuto prima di entrare nello stato desiderato.

before (callable)

Una funzione PHP richiamabile che viene richiamata prima di ogni tentativo. La funzione richiamabile viene richiamata con il comando `Aws\CommandInterface` che sta per essere eseguito e il numero di tentativi eseguiti fino a ora. L'utilizzo della funzione richiamabile `before` potrebbe modificare i comandi prima che vengano eseguiti oppure fornire le informazioni sullo stato di avanzamento.

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

Attesa in modo asincrono

Oltre ad attendere in modo sincrono, è possibile richiamare un waiter per l'attesa in modo asincrono durante l'invio di altre richieste oppure attendere più risorse contemporaneamente.

È possibile accedere a una promessa di waiter recuperando un waiter da un client con il metodo `getWaiter($name, array $args = [])` del client. Utilizza il metodo `promise()` di un waiter per inizializzare il waiter. Una promessa da parte di un waiter viene soddisfatta con l'ultimo `Aws\CommandInterface` che è stato eseguito nel waiter e rifiutata con un `RuntimeException` in caso di errore.

```
use Aws\CommandInterface;

$waiterName = 'BucketExists';
$waiterOptions = ['Bucket' => 'amzn-s3-demo-bucket'];

// Create a waiter promise
```

```
$waiter = $s3Client->getWaiter($waiterName, $waiterOptions);

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

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

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

L'esposizione di una API di waiter basati su promessa consente alcuni casi d'uso potenti e con costi di gestione relativamente bassi. Ad esempio, cosa fare se si desidera attendere più risorse e si interviene con il primo waiter che si è risolto correttamente?

```
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 espressioni nella AWS SDK per PHP versione 3

JMESPathconsente di specificare in modo dichiarativo come estrarre elementi da un documento JSON. Dipende da jmespath.php per alimentare alcune delle astrazioni di alto livello come Paginators nella AWS SDK per PHP versione 3 e Waiters nella versione 3, ma AWS SDK per PHP espone anche la AWS SDK per PHP ricerca su e. JMESPath `Aws\ResultInterface` `Aws\ResultPaginator`

Puoi giocare con il tuo browser provando gli esempi online JMESPath . JMESPath Puoi saperne di più sul linguaggio, comprese le espressioni e le funzioni disponibili, nelle JMESPath specifiche.

I AWS CLIsupporti JMESPath. Le espressioni create per l'output CLI sono del tutto compatibili con le espressioni create per AWS SDK per PHP.

Estrazione di dati dai risultati

L'`Aws\ResultInterface`interfaccia ha un `search($expression)` metodo che estrae i dati da un modello di risultato basato su un' JMESPath espressione. L'uso di JMESPath espressioni per interrogare i dati da un oggetto risultato può aiutare a rimuovere il codice condizionale standard e a esprimere in modo più conciso i dati che vengono estratti.

Per dimostrarne il funzionamento, inizieremo con l'output JSON predefinito riportato di seguito, che descrive due volumi Amazon Elastic Block Store (Amazon EBS) collegati a istanze Amazon separate. 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"
        }
    }
}
```

Innanzitutto, è possibile recuperare solo il primo volume dall'elenco dei volumi con il comando seguente.

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

Quindi, viene utilizzata l'espressione `wildcard-index [*]` per la reiterazione nell'intero elenco e, inoltre, vengono estratti e rinominati tre elementi: `VolumeId` viene rinominato in `ID`, `AvailabilityZone` viene rinominato in `AZ` e `Size` resta `Size`. È possibile estrarre e rinominare questi elementi utilizzando un'espressione `multi-hash` posta dopo l'espressione `wildcard-index`.

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

Ciò offre una serie di dati PHP come i seguenti:

```
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)
  }
}
```

Nella notazione `multi-hash` è possibile anche usare chiavi concatenate, ad esempio `key1.key2[0].key3` , per estrarre elementi profondamente nidificati all'interno della struttura. L'esempio di seguito mostra questo tipo di utilizzo con la chiave `Attachments[0].InstanceId`

assegnata semplicemente in alias a `InstanceId`. (Nella maggior parte dei casi, JMESPath le espressioni ignoreranno gli spazi bianchi).

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

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

L'espressione precedente genererà i seguenti dati:

```
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)
  }
}
```

È inoltre possibile filtrare più elementi con l'espressione `multi-list:[key1, key2]`. In questo modo verranno formattati tutti gli attributi filtrati in un unico elenco ordinato per ogni oggetto, indipendentemente dal tipo.

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

L'esecuzione della ricerca precedente produce i seguenti dati:

```
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)
  }
}
```

Utilizza un'espressione `filter` per filtrare i risultati in base al valore di un campo specifico. Il seguente esempio genera soltanto volumi nella `us-west-2a` zona di disponibilità.

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

JMESPath supporta anche le espressioni di funzioni. Supponiamo di voler eseguire la stessa query di cui sopra, ma recuperare invece tutti i volumi in cui il volume si trova in una AWS regione che inizia con «us-». La seguente espressione utilizza la funzione `starts_with` e passa una stringa letterale di `us-`. Il risultato di questa funzione viene quindi confrontato con il valore letterale JSON `true`, passando solo i risultati del predicato del filtro che ha restituito `true` mediante la proiezione del filtro.

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

Estrazione di dati dagli impaginatori

Come sapete dai Paginators della guida alla AWS SDK per PHP versione 3, `Aws\ResultPaginator` gli oggetti vengono utilizzati per ottenere risultati da un'operazione API paginabile. AWS SDK per PHP Consente di estrarre e iterare dati filtrati dagli `Aws\ResultPaginator` oggetti, essenzialmente implementando una mappa piatta sull'iteratore in cui il risultato di un'espressione è la funzione di mappa. JMESPath

Supponiamo di volere creare una funzione `iterator` che produca solo oggetti da un bucket di dimensioni superiori a 1 MB. Questo risultato può essere ottenuto creando innanzitutto un impaginatore `ListObjects`, quindi applicando ad esso una funzione `search()` e creando un'iterazione di tipo flatmap sui dati impaginati.

```
$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);
}
```

Usa l'estensione AWS Common Runtime (AWS CRT)

Le librerie AWS CRT forniscono funzionalità di base con buone prestazioni e un ingombro minimo per diverse applicazioni. AWS SDKs Questo argomento descrive quando il AWS CRT viene utilizzato dall'SDK for PHP e come installare l'estensione CRT. AWS

Quando è necessario installare l'estensione CRT AWS

L'SDK for PHP utilizza le funzionalità AWS di autorizzazione e checksum delle librerie CRT. L'estensione AWS CRT è necessaria quando si lavora con:

- Punti di accesso multi-Regione di Amazon S3
- Endpoint EventBridge globali Amazon
- Un algoritmo di checksum CRC-32C in Amazon Simple Storage Service (Amazon S3)

Se utilizzi una delle funzionalità sopra elencate e l'estensione AWS CRT non è installata nel tuo ambiente PHP, l'SDK for PHP segnalerà un messaggio di errore e ti ricorderà di installare l'estensione.

Installa l'estensione AWS Common Runtime (CRT)AWS

Le istruzioni su come installare l'estensione AWS CRT sono disponibili nella pagina principale del GitHubrepository di. aws-crt-php

Aggiornamento dalla versione 2 di AWS SDK per PHP

Questo argomento mostra come migrare il codice per utilizzare la versione 3 dell' AWS SDK per PHP e come la nuova versione differisce dalla versione 2 dell'SDK.

> **Note**
>
> Il modello di utilizzo di base dell'SDK (ad esempio, `$result = $client->operation($params);`) non ha subito modifiche dalla versione 2 alla versione 3, pertanto la migrazione dovrebbe risultare agevole.

Introduzione

La versione 3 di AWS SDK per PHP rappresenta uno sforzo significativo per migliorare le funzionalità dell'SDK, incorporare oltre due anni di feedback dei clienti, aggiornare le nostre dipendenze, migliorare le prestazioni e adottare gli standard PHP più recenti.

Novità della versione 3

La versione 3 AWS SDK per PHP segue gli standard PSR-4 e PSR-7 e seguirà lo standard in futuro. SemVer

Altre nuove caratteristiche includono

- Sistema middleware per la personalizzazione del comportamento del client di servizio
- Impaginatori flessibili per navigare attraverso i risultati impaginati
- Capacità di interrogare i dati dagli oggetti dei risultati e degli impaginatori con JMESPath

- Debug semplice tramite l'opzione di configurazione `'debug'`

Livello HTTP disassociato

- Guzzle 6 viene utilizzato per impostazione predefinita per l'invio di richieste, ma è supportato anche Guzzle 5.
- L'SDK funzionerà in ambienti in cui cURL non è disponibile.
- Sono supportati anche i gestori HTTP personalizzati.

Richieste asincrone

- Caratteristiche come waiters e uploader in più parti possono essere utilizzate anche in modo asincrono.
- I flussi di lavoro asincroni possono essere creati utilizzando promesse e coroutine.
- Le prestazioni delle richieste simultanee o in batch sono migliorate.

Cosa cambia rispetto alla versione 2

Le dipendenze di progetto sono aggiornate

Le dipendenze dell'SDK sono state modificate in questa versione.

- L'SDK ora richiede PHP 5.5+. Utilizziamo generatori liberamente all'interno del codice SDK.
- Abbiamo aggiornato l'SDK per utilizzare Guzzle 6 (o 5), che fornisce l'implementazione del client HTTP sottostante utilizzata dall'SDK per inviare richieste ai servizi. AWS La versione più recente di Guzzle offre una serie di miglioramenti, tra cui richieste asincrone, gestori HTTP commutabili, conformità PSR-7, prestazioni migliori e molto altro.
- Il pacchetto PSR-7 di PHP-FIG (`psr/http-message`) definisce le interfacce per la rappresentazione di richieste HTTP, risposte HTTP e flussi. URLs Queste interfacce vengono nell'SDK e in Guzzle, che offre interoperabilità con altri pacchetti conformi a PSR-7.
- L'implementazione PSR-7 di Guzzle (`guzzlehttp/psr7`) fornisce un'implementazione delle interfacce in PSR-7 e diverse utili classi e funzioni. Sia l'SDK che Guzzle 6 fanno molto affidamento su questo pacchetto.
- L'implementazione di Promesse/A+ di Guzzle (`guzzlehttp/promises`) viene utilizzata nell'SDK e in Guzzle per fornire interfacce per la gestione delle richieste e delle coroutine asincrone. Mentre

il gestore HTTP multi-cURL di Guzzle in ultima analisi implementa il modello di I/O senza blocchi che consente le richieste asincrone, questo pacchetto offre la possibilità di programmare all'interno di tale paradigma. Vedi Promises nella versione 3 per maggiori dettagli. AWS SDK per PHP

- L'implementazione PHP di JMESPath(`mtdowling/jmespath.php`) viene utilizzata nell'SDK per fornire la capacità di interrogazione dei dati dei metodi and. `Aws\Result::search()` `Aws\ResultPaginator::search()` Vedi JMESPath Expressions nella AWS SDK per PHP versione 3 per maggiori dettagli.

Le opzioni di regione e versione (Region e Version) sono ora obbligatorie

Quando si creano istanze per un client per qualsiasi servizio, specificare le opzioni `'region'` e `'version'`. Nella versione 2 di AWS SDK per PHP, `'version'` era completamente opzionale e talvolta `'region'` era facoltativo. Nella versione 3, entrambe le opzioni sono sempre obbligatorie. Essere espliciti su entrambe queste opzioni ti consente di concentrarti sulla versione dell'API e sulla AWS regione in cui stai codificando. Quando vengono create nuove versioni dell'API o diventano disponibili nuove AWS regioni, sarai isolato da modifiche potenzialmente irreversibili finché non sarai pronto ad aggiornare esplicitamente la tua configurazione.

> **Note**
>
> Se non sei interessato alla versione API che usi, puoi semplicemente impostare l'opzione `'version'` su `'latest'`. Tuttavia, ti consigliamo di impostare i numeri di versione dell'API in modo esplicito per il codice di produzione.
> Non tutti i servizi sono disponibili in tutte le AWS regioni. Un elenco delle regioni disponibili è presente nel riferimento Regioni ed endpoint.
> Per i servizi disponibili solo tramite un singolo endpoint globale (ad esempio Amazon Route 53 e Amazon CloudFront) AWS Identity and Access Management, crea un'istanza dei client con la regione configurata impostata su. `us-east-1`

> **Important**
>
> L'SDK include anche client multiregionali, che possono inviare richieste a diverse AWS regioni in base a un parametro (`@region`) fornito come parametro di comando. La regione utilizzata per impostazione predefinita da questi client è specificata con l'opzione `region` fornita al costruttore del client.

L'istanza del client usa il costruttore

Nella versione 3 di AWS SDK per PHP, il modo in cui si crea un'istanza di un client è cambiato. Anziché utilizzare i metodi `factory` nella versione 2, è possibile creare con facilità un'istanza di un client utilizzando la parola chiave `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**
>
> La creazione dell'istanza di un client utilizzando il metodo `factory()` funziona ancora. Tuttavia, questa azione è considerata obsoleta.

La configurazione del client è stata modificata

Le opzioni di configurazione del client nella versione 3 di sono leggermente AWS SDK per PHP cambiate rispetto alla versione 2. Vedi la pagina Configurazione per la AWS SDK per PHP versione 3 per una descrizione di tutte le opzioni supportate.

> **Important**
>
> Nella versione 3 `'key'` e `'secret'` non sono più opzioni valide al livello radice, ma è possibile passarle come parte dell'opzione `'credentials'`. Uno dei motivi per cui l'abbiamo fatto è stato quello di scoraggiare gli sviluppatori dall'inserire le proprie AWS credenziali nei loro progetti.

L'oggetto Sdk

La versione 3 di AWS SDK per PHP introduce l'oggetto in sostituzione di`Aws\Sdk`. `Aws\Common` `\Aws` L'oggetto `Sdk` agisce come un client factory e viene utilizzato per gestire le opzioni di configurazione condivise da più client.

Anche se la classe `Aws` nella versione 2 dell'SDK funzionava come localizzatore di servizio (restituiva sempre la stessa istanza di un client), la classe `Sdk` nella versione 3 restituisce una nuova istanza di un client ogni volta che viene utilizzata.

L'oggetto `Sdk`, inoltre, non supporta lo stesso formato di file di configurazione dalla versione 2 dell'SDK. Questo formato di configurazione era specifico per Guzzle 3 ed è ora obsoleto. La configurazione può essere eseguita più semplicemente con gli array di base ed è documentata in Utilizzo della classe Sdk.

Alcuni risultati delle API sono cambiati

Per garantire la coerenza nel modo in cui l'SDK analizza il risultato di un'operazione API, Amazon, ElastiCache Amazon RDS e Amazon Redshift ora dispongono di un elemento di wrapping aggiuntivo su alcune risposte API.

Ad esempio, la chiamata al DescribeEngineDefaultParametersrisultato di Amazon RDS nella versione 3 ora include un elemento wrapping «EngineDefaults». Nella versione 2, questo elemento non era presente.

```
$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'];
```

Le seguenti operazioni sono interessate e ora contengono un elemento di wrapping nell'output del risultato (fornito di seguito tra parentesi):

- Amazon ElastiCache
  - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
  - CopySnapshot (Istantanea)
  - CreateCacheCluster (CacheCluster)
  - CreateCacheParameterGroup (CacheParameterGroup)
  - CreateCacheSecurityGroup (CacheSecurityGroup)
  - CreateCacheSubnetGroup (CacheSubnetGroup)
  - CreateReplicationGroup (ReplicationGroup)
  - CreateSnapshot (Istantanea)
  - DeleteCacheCluster (CacheCluster)
  - DeleteReplicationGroup (ReplicationGroup)
  - DeleteSnapshot (Istantanea)
  - DescribeEngineDefaultParameters (EngineDefaults)
  - ModifyCacheCluster (CacheCluster)
  - ModifyCacheSubnetGroup (CacheSubnetGroup)
  - ModifyReplicationGroup (ReplicationGroup)
  - PurchaseReservedCacheNodesOffering (ReservedCacheNode)
  - RebootCacheCluster (CacheCluster)
  - RevokeCacheSecurityGroupIngress (CacheSecurityGroup)
- Amazon RDS
  - AddSourceIdentifierToSubscription (EventSubscription)
  - Autorizza DBSecurity GroupIngress (gruppo) DBSecurity
  - Copia DBParameter gruppo (DBParametergruppo)
  - Copia DBSnapshot (DBSnapshot)
  - CopyOptionGroup (OptionGroup)
  - Crea DBInstance (DBInstance)
  - Crea DBInstance ReadReplica (DBInstance)
  - Crea DBParameter gruppo (DBParametergruppo)

- Crea DBSecurity gruppo (DBSecuritygruppo)
- Crea DBSnapshot (DBSnapshot)
- Crea DBSubnet gruppo (DBSubnetgruppo)
- CreateEventSubscription (EventSubscription)
- CreateOptionGroup (OptionGroup)
- Elimina DBInstance (DBInstance)
- Elimina DBSnapshot (DBSnapshot)
- DeleteEventSubscription (EventSubscription)
- DescribeEngineDefaultParameters (EngineDefaults)
- Modifica DBInstance (DBInstance)
- Modifica DBSubnet gruppo (DBSubnetgruppo)
- ModifyEventSubscription (EventSubscription)
- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesOfferta (riservataDBInstance)
- Riavvio DBInstance () DBInstance
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- Ripristina DBInstance da DBSnapshot () DBInstance
- Ripristina DBInstance ToPointInTime (DBInstance)
- Revoca DBSecurity GroupIngress (gruppo) DBSecurity

- Amazon Redshift
  - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
  - AuthorizeSnapshotAccess (Istantanea)
  - CopyClusterSnapshot (Istantanea)
  - CreateCluster (Cluster)
  - CreateClusterParameterGroup (ClusterParameterGroup)
  - CreateClusterSecurityGroup (ClusterSecurityGroup)
  - CreateClusterSnapshot (Istantanea)
  - CreateClusterSubnetGroup (ClusterSubnetGroup)

- CreateEventSubscription (EventSubscription)

- CreateHsmClientCertificate (HsmClientCertificate)
- CreateHsmConfiguration (HsmConfiguration)
- DeleteCluster (Cluster)
- DeleteClusterSnapshot (Istantanea)
- DescribeDefaultClusterParameters (DefaultClusterParameters)
- DisableSnapshotCopy (Cluster)
- EnableSnapshotCopy (Grappolo)
- ModifyCluster (Grappolo)
- ModifyClusterSubnetGroup (ClusterSubnetGroup)
- ModifyEventSubscription (EventSubscription)
- ModifySnapshotCopyRetentionPeriod (Grappolo)
- PurchaseReservedNodeOffering (ReservedNode)
- RebootCluster (Grappolo)
- RestoreFromClusterSnapshot (Grappolo)
- RevokeClusterSecurityGroupIngress (ClusterSecurityGroup)
- RevokeSnapshotAccess (Istantanea)
- RotateEncryptionKey (Cluster)

Classi di enumerazione che sono state rimosse

Abbiamo rimosso le classi `Enum` (ad esempio, `Aws\S3\Enum\CannedAcl`) presenti nella versione 2 dell' AWS SDK per PHP. Le enumerazioni erano classi concrete all'interno dell'API pubblica dell'SDK che conteneva costanti che rappresentano gruppi di validità i valori dei parametri. Poiché queste enumerazioni sono specifiche per le versioni dell'API, possono cambiare nel tempo, possono entrare in conflitto con parole riservate PHP e non sono risultate molto utili, le abbiamo rimosse nella versione 3. Questo supporta la natura agnostica della versione API basato su dati della versione 3.

Invece di utilizzare i valori di oggetti `Enum`, utilizza i valori letterali direttamente (ad esempio `CannedAcl::PUBLIC_READ` → `'public-read'`).

Classi di eccezioni granulari sono state rimosse

Abbiamo rimosso le classi di eccezione granulari esistenti negli spazi dei nomi di ciascun servizio (ad esempio, `Aws\Rds\Exception\{SpecificError}Exception`) per motivi molto simili a quelli per

cui abbiamo rimosso le enumerazioni. Le eccezioni generate da un servizio o operazione dipendono da quale versione dell'API viene utilizzata (possono variare da una versione all'altra). Inoltre, l'elenco completo delle eccezioni che possono essere generate da una determinata operazione non è disponibile, il che rendeva le classi di eccezione granulari della versione 2 incomplete.

Gestire gli errori ricevendo la classe di eccezione root per ciascun servizio (ad esempio, `Aws\Rds\Exception\RdsException`). È possibile utilizzare il metodo di eccezione `getAwsErrorCode()` per verificare la presenza di codici di errore specifici. Questo è funzionalmente equivalente a catturare diverse classi di eccezioni, ma prevede che funzionino senza aggiungere dimensioni all'SDK.

Le classi Facade statiche sono state rimosse

Nella versione 2 di AWS SDK per PHP, c'era una funzionalità oscura ispirata a Laravel che permetteva di `enableFacades()` richiamare la `Aws` classe per abilitare l'accesso statico ai vari client di servizio. Questa caratteristica va contro le best practice di PHP e abbiamo smesso di documentarla più di un anno fa. Nella versione 3, questa caratteristica è stata completamente rimossa. Recuperare gli oggetti client dall'oggetto `Aws\Sdk` e utilizzarli come istanze di oggetti, non classi statiche.

Gli impaginatori sostituiscono gli iteratori

La versione 2 di AWS SDK per PHP aveva una funzionalità denominata * iteratori*. Questi oggetti sono stati utilizzati per reiterare i risultati impaginati in modo eccessivo. Un reclamo che abbiamo ricevuto su questi oggetti era che non erano sufficientemente flessibili, perché l'iteratore emetteva solo valori specifici da ciascun risultato. Se avevi bisogno di altri valori dai risultati, potevi recuperarli solo tramite listener di eventi.

Nella versione 3, gli iteratori sono stati sostituiti con Impaginatori. Lo scopo è simile, ma gli impaginatori sono più flessibili. Questo è dovuto al fatto che restituiscono gli oggetti di risultato anziché i valori da una risposta.

I seguenti esempi mostrano come gli impaginatori differiscono dagli iteratori, dimostrando come recuperare i risultati impaginati per l'operazione `S3 ListObjects` sia nella versione 2 che nella versione 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";
    }
}
```

Gli oggetti Paginator dispongono di un `search()` metodo che consente di utilizzare JMESPathle espressioni per estrarre più facilmente i dati dal set di risultati.

```
$results = $s3Client->getPaginator('ListObjects', ['Bucket' => 'amzn-s3-demo-bucket']);
foreach ($results->search('Contents[].Key') as $key) {
    echo $key . "\n";
}
```

> **Note**
>
> Il metodo `getIterator()` è ancora supportato per consentire una transizione agevole alla versione 3, ma ti consigliamo di migrare il codice per l'utilizzo degli impaginatori.

Molte astrazioni di livello più elevato sono cambiate

In generale, molte delle astrazioni di livello più elevato (oggetti helper specifici per il servizio, tranne i client) sono state migliorate o aggiornate. Alcune sono state rimosse.

- Aggiornato:
  - Il modo in cui usi il caricamento in più parti di Amazon S3 è cambiato. Amazon S3 Glacier Multipart Upload è stato modificato in modo simile.
  - Il modo di creare Amazon S3 prefirmato URLs è cambiato.
  - Lo spazio dei nomi `Aws\S3\Sync` è stato sostituito dalla classe `Aws\S3\Transfer`. I metodi `S3Client::uploadDirectory()` e `S3Client::downloadBucket()` sono ancora disponibili, ma hanno opzioni diverse. Consulta la documentazione per Amazon S3 Transfer Manager con AWS SDK per PHP versione 3.

    - `Aws\S3\Model\ClearBucket` e `Aws\S3\Model\DeleteObjectsBatch` sono stati sostituiti da `Aws\S3\BatchDelete` e `S3Client::deleteMatchingObjects()`.
    - Le opzioni e i comportamenti per Using the DynamoDB Session Handler AWS SDK per PHP con la versione 3 sono leggermente cambiati.
    - Lo spazio dei nomi `Aws\DynamoDb\Model\BatchRequest` è stato sostituito da `Aws\DynamoDb\WriteRequestBatch`. Consulta la documentazione per DynamoDB. WriteRequestBatch
    - `Aws\Ses\SesClient` gestisce ora la codifica base64 di `RawMessage` quando si usa l'operazione `SendRawEmail`.
- Rimosso>
    - Amazon `Item` DynamoDB `AttributeItemIterator` e classi: erano precedentemente obsolete nella versione 2.7.0.
    - Amazon SNS Message Validator: ora si tratta di un progetto separato e leggero che non richiede l'SDK come dipendenza. Questo progetto è tuttavia incluso nelle distribuzioni Phar e ZIP dell'SDK. Puoi trovare una guida introduttiva sul blog PHP Development. AWS
    - Amazon S3 `AcpBuilder` e gli oggetti correlati sono stati rimossi.

Confronto tra gli esempi di codice da parte di entrambe le versioni dell'SDK

Gli esempi seguenti mostrano alcuni dei modi in cui l'utilizzo della versione 3 di AWS SDK per PHP potrebbe differire dalla versione 2.

Esempio: funzionamento di Amazon S3 ListObjects

Dalla versione 2 dell'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";
}
```

Dalla versione 3 dell'SDK

Differenze principali:

- Usa `new` invece di `factory()` per creare un'istanza del client.
- Le opzioni `'version'` e `'region'` sono obbligatorie durante la creazione dell'istanza.

```
<?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";
}
```

Esempio: creazione di un'istanza di un client con configurazione globale

Dalla versione 2 dell'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.
```

Dalla versione 3 dell'SDK

Differenze principali:

- Utilizza la classe `Aws\Sdk` anziché `Aws\Common\Aws`.
- Non c'è un file di configurazione. Utilizza un array per la configurazione.
- L'opzione `'version'` è obbligatoria durante la creazione dell'istanza.
- Utilizza i metodi `create<Service>()` anziché `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.
```

Condivisi **`config`** e file **`credentials`**

I `credentials` file AWS `config` condivisi sono il modo più comune per specificare l'autenticazione e la configurazione per AWS SDK per PHP. Utilizza questi file per archiviare le impostazioni che gli strumenti e le applicazioni possono utilizzare in tutto il AWS SDKs mondo AWS Command Line Interface.

I file condivisi AWS `config` e sono `credentials` file di testo semplice che si trovano per impostazione predefinita in una cartella denominata `.aws` che si trova nella cartella "home" del computer. Per informazioni dettagliate sulla posizione di questi file, consulta Posizione dei file

condivisi `config` e `credentials` condivisi nella Guida di riferimento agli strumenti AWS SDKs e strumenti.

Per tutte le impostazioni che è possibile memorizzare in questi file, consulta il riferimento alle impostazioni di configurazione AWS SDKs e autenticazione nella Guida di riferimento agli strumenti. Questo riferimento copre anche la precedenza nell'applicazione di impostazioni da fonti alternative come le variabili di ambiente.

Profili denominati

Le impostazioni all'interno dei `credentials` file `config` condivisi sono associate a un profilo specifico. Con più profili, puoi creare diverse configurazioni di impostazioni da applicare in diversi scenari. Uno dei profili è designato come `default` profilo e viene utilizzato automaticamente quando non si specifica esplicitamente un profilo da utilizzare.

Per ulteriori informazioni sulla configurazione di profili denominati, consulta `config`Shared and `credentials` files nella AWS SDKs and Tools Reference Guide.

È possibile specificare un profilo denominato da utilizzare per creare un'istanza di un client utilizzando l'`profile`opzione:

```
use Aws\DynamoDb\DynamoDbClient;

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

Lavora con AWS i servizi in AWS SDK per PHP

Le sezioni seguenti contengono esempi, tutorial, attività e guide che mostrano come utilizzarli per AWS SDK per PHP lavorare con AWS i servizi.

Argomenti

- Usa le funzionalità e le opzioni della AWS SDK per PHP versione 3
- Esempi di codice con linee guida per AWS SDK per PHP

Usa le funzionalità e le opzioni della AWS SDK per PHP versione 3

La AWS SDK per PHP versione 3 fornisce supporto per funzionalità e opzioni aggiuntive con cui lavorare Servizio AWS APIs. Le sezioni di questo argomento trattano queste opzioni per servizio.

Argomenti

- Utilizzo del gestore di sessione DynamoDB con la versione 3 AWS SDK per PHP
- Caratteristiche e opzioni di Amazon S3

Utilizzo del gestore di sessione DynamoDB con la versione 3 AWS SDK per PHP

Il DynamoDB Session Handler è un gestore di sessioni personalizzato per PHP che consente agli sviluppatori di utilizzare Amazon DynamoDB come archivio di sessioni. L'utilizzo di DynamoDB per l'archiviazione delle sessioni allevia i problemi che si verificano con la gestione delle sessioni in un'applicazione Web distribuita spostando le sessioni dal file system locale a una posizione condivisa. DynamoDB è veloce, scalabile, facile da configurare e gestisce automaticamente la replica dei dati.

Il DynamoDB Session Handler utilizza la `session_set_save_handler()` funzione per agganciare le operazioni DynamoDB alle funzioni di sessione native di PHP per consentire un vero calo delle sostituzioni. È incluso il supporto per caratteristiche quali blocco delle sessioni e garbage collection, che fanno parte del gestore di sessione predefinito di PHP.

Per ulteriori informazioni sul servizio DynamoDB, consulta la home page di Amazon DynamoDB.

Utilizzo di base

Passaggio 1: registrare il gestore

Per prima cosa, crea istanze e registra il gestore di sessione.

```
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();
```

Fase 2: Crea una tabella per archiviare le tue sessioni

Prima di poter utilizzare il gestore di sessione devi creare una tabella in cui archiviare le sessioni. Puoi farlo in anticipo utilizzando la AWS Console per Amazon DynamoDB o utilizzando il. AWS SDK per PHP

Quando crei questa tabella utilizza "id" come nome della chiave primaria. Inoltre è consigliabile configurare un attributo Time To Live utilizzando l'attributo "expires" per trarre vantaggio dalla garbage collection automatica di sessioni.

Fase 3. Usa le sessioni PHP come faresti normalmente

Una volta che il gestore di sessione è stato registrato e la tabella è stata creata, è possibile eseguire operazioni di lettura e scrittura con la sessione utilizzando la variabile superglobale `$_SESSION`, in modo analogo a quanto avviene normalmente con il gestore di sessione predefinito di PHP. Il DynamoDB Session Handler incapsula e riassume le interazioni con DynamoDB e consente di utilizzare semplicemente le funzioni e l'interfaccia di sessione native di PHP.

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

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

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

Configurazione

È possibile configurare il comportamento del gestore di sessione utilizzando le seguenti opzioni. Tutte le opzioni sono facoltative, ma assicurati di comprendere quali sono le impostazioni predefinite.

`table_name`

Il nome della tabella DynamoDB in cui archiviare le sessioni. L'impostazione predefinita di questa opzione è `'sessions'`.

`hash_key`

Il nome della chiave hash nella tabella delle sessioni di DynamoDB. L'impostazione predefinita di questa opzione è `'id'`.

`data_attribute`

Il nome dell'attributo nella tabella delle sessioni di DynamoDB in cui sono archiviati i dati della sessione. L'impostazione predefinita di questa opzione è `'data'`.

`data_attribute_type`

Il tipo di attributo nella tabella delle sessioni di DynamoDB in cui sono archiviati i dati della sessione. L'impostazione predefinita di questa opzione è `'string'`, ma può essere facoltativamente impostato su `'binary'`.

`session_lifetime`

La durata di una sessione inattiva prima che venga sottoposta al processo di garbage collection. Se non viene fornito, il valore della durata che verrà utilizzato è `ini_get('session.gc_maxlifetime')`.

`session_lifetime_attribute`

Il nome dell'attributo nella tabella delle sessioni di DynamoDB in cui è memorizzata l'ora di scadenza della sessione. L'impostazione predefinita di questa opzione è `'expires'`.

`consistent_read`

Se il gestore di sessione dovrebbe utilizzare letture consistenti per l'operazione `GetItem`. Il valore predefinito è `true`.

`locking`

Se utilizzare il blocco delle sessioni. Il valore predefinito è `false`.

`batch_config`

Configurazione utilizzata per le eliminazioni in batch durante il processo di garbage collection. Queste opzioni vengono passate direttamente agli oggetti WriteRequestBatchDynamoDB. Attiva manualmente il processo di garbage collection tramite `SessionHandler::garbageCollect()`.

`max_lock_wait_time`

Tempo massimo (in secondi) che il gestore di sessione dovrebbe attendere per acquisire un blocco prima di desistere. L'impostazione predefinita per questa opzione è `10` e viene utilizzata solo con il blocco della sessione.

`min_lock_retry_microtime`

Tempo minimo (in microsecondi) che il gestore di sessione dovrebbe attendere tra i tentativi di acquisire un blocco. L'impostazione predefinita per questa opzione è `10000` e viene utilizzata solo con il blocco della sessione.

`max_lock_retry_microtime`

Tempo massimo (in microsecondi) che il gestore di sessione dovrebbe attendere tra i tentativi di acquisire un blocco. L'impostazione predefinita per questa opzione è `50000` e viene utilizzata solo con il blocco della sessione.

Per configurare il Gestore di sessione, specifica le opzioni di configurazione durante la creazione di istanze per il gestore. Il codice seguente è un esempio con tutte le opzioni di configurazione specificate.

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

Prezzi

Oltre ai costi di archiviazione e trasferimento dei dati, i costi associati all'utilizzo di DynamoDB vengono calcolati in base alla capacità di throughput fornita della tabella (consulta i dettagli sui prezzi di Amazon DynamoDB). Il throughput viene misurato in unità di capacità di scrittura e lettura. La home page di Amazon DynamoDB dice:

Un'unità di capacità di lettura equivale a una lettura fortemente consistente al secondo (o a due letture consistenti finali al secondo) per elementi di dimensioni fino a 4 KB. Un'unità di capacità di scrittura equivale a una scrittura al secondo per elementi di dimensioni fino a 1 KB.

In ultima analisi, il throughput e i costi richiesti per la tabella delle sessioni saranno correlati alle previsioni del traffico e delle dimensioni delle sessioni. La tabella seguente spiega la quantità di operazioni di lettura e scrittura eseguite sulla tabella DynamoDB per ciascuna delle funzioni di sessione.

Lettura tramite `session_start()`	• 1 operazione di lettura (solo 0,5 se `consistent_read` è `false`). • (Condizionale) 1 operazione di scrittura per eliminare la sessione se è scaduta.
Lettura tramite `session_start()` (utilizzando il blocco della sessione)	• Un minimo di 1 operazione di scrittura. • (Condizionale) Ulteriori operazioni di scrittura per ogni tentativo di acquisizione di un blocco della sessione. Sulla base di un tempo di attesa per il blocco configurato e di opzioni per nuovi tentativi. • (Condizionale) 1 operazione di scrittura per eliminare la sessione se è scaduta.
Scrittura tramite `session_write_close()`	• 1 operazione di scrittura.
Eliminazione tramite `session_destroy()`	• 1 operazione di scrittura.

Garbage Collection	• 0,5 operazioni di lettura per 4 KB di dati nella tabella per rilevare sessioni scadute. • 1 operazione di scrittura per elemento scaduto per eliminarlo.

Blocco della sessione

Il gestore di sessione DynamoDB supporta il blocco pessimistico delle sessioni per imitare il comportamento del gestore di sessione predefinito di PHP. Per impostazione predefinita, il DynamoDB Session Handler ha questa funzionalità disattivata perché può diventare un ostacolo alle prestazioni e far aumentare i costi, specialmente quando un'applicazione accede alla sessione quando utilizza richieste Ajax o iframe. Valuta attentamente se l'applicazione richiede il blocco della sessione prima di abilitarla.

Per abilitare il blocco della sessione, imposta l'opzione `'locking'` su `true` quando crei istanze per `SessionHandler`.

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

Raccolta dei rifiuti

Configura un attributo TTL nella tabella DynamoDB utilizzando l'attributo "expires". In questo modo, il processo garbage collection delle sessioni verrà eseguito automaticamente.

In alternativa, il gestore di sessione DynamoDB supporta la raccolta dei rifiuti di sessione utilizzando una serie di operazioni and. `Scan` `BatchWriteItem` Per via della natura dell'operazione `Scan` e al fine di trovare tutte le sessioni scadute e rimuoverle, il processo di garbage collection può richiedere una notevole capacità di throughput assegnata.

Per questo motivo, la garbage collection automatizzata non è supportata. Si consiglia di pianificare il processo di garbage collection durante gli orari meno intensi, in modo che l'incremento di throughput non abbia ripercussioni sul resto dell'applicazione. Ad esempio, si potrebbe utilizzare un processo cron notturno che attivi uno script per l'esecuzione del processo di garbage collection. Il contenuto dello script dovrebbe essere simile a quello riportato di seguito.

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

$sessionHandler->garbageCollect();
```

È inoltre possibile utilizzare l'opzione `'before'` all'interno di `'batch_config'` per includere ritardi sulle operazioni `BatchWriteItem` eseguite dal processo di garbage collection. Ciò aumenterà il tempo necessario per completare la raccolta dei rifiuti, ma può aiutarti a distribuire le richieste fatte dal gestore di sessione DynamoDB per aiutarti a rimanere vicino o entro la capacità di throughput assegnata durante la raccolta dei rifiuti.

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

$sessionHandler->garbageCollect();
```

Best practice

1. Crea la tabella delle sessioni in una AWS regione geograficamente più vicina o nella stessa regione dei tuoi server delle applicazioni. Ciò garantisce la latenza più bassa tra l'applicazione e il database DynamoDB.
2. Scegli attentamente la capacità di throughput assegnata della tabella delle sessioni. Tieni in considerazione il traffico previsto verso la tua applicazione e le probabili dimensioni delle sessioni. In alternativa, utilizza la modalità di capacità in lettura/scrittura "on demand" per la tabella.

3. Monitora il throughput consumato tramite la Console di AWS gestione o con Amazon CloudWatch e modifica le impostazioni di throughput secondo necessità per soddisfare le esigenze della tua applicazione.
4. Mantieni le dimensioni delle sessioni ridotte (preferibilmente inferiori a 1 KB). Sessioni di dimensioni ridotte garantiscono prestazioni migliori e richiedono una capacità di throughput assegnata inferiore.
5. Non utilizzare il blocco delle sessioni, a meno che la tua applicazione non lo richieda.
6. Invece di utilizzare trigger per la garbage collection delle sessioni integrati in PHP, pianifica la garbage collection utilizzando un processo cron o un altro meccanismo di pianificazione in modo che venga eseguita durante gli orari meno intensi. Sfrutta i vantaggi dell'opzione `'batch_config'`.

Autorizzazioni IAM richieste

Per utilizzare SessionHhandler DynamoDB, le credenziali configurate devono disporre dell'autorizzazione per utilizzare la tabella DynamoDB creata in un passaggio precedente. La seguente policy IAM contiene le autorizzazioni minime necessarie. Per utilizzare questa politica, sostituisci il valore Resource con l'Amazon Resource Name (ARN) della tabella che hai creato in precedenza. Per ulteriori informazioni sulla creazione e l'associazione delle policy IAM, consulta Managing IAM Policies nella IAM User Guide.

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

Caratteristiche e opzioni di Amazon S3

In questo argomento vengono illustrate le funzionalità e le opzioni aggiuntive fornite dalla AWS SDK per PHP versione 3 per l'utilizzo con Amazon S3.

Argomenti

- Client multiregionale Amazon S3 con versione 3 AWS SDK per PHP
- Stream wrapper Amazon S3 con versione 3 AWS SDK per PHP
- Gestione trasferimenti Amazon S3 con AWS SDK per PHP versione 3
- Crittografia lato client di Amazon S3 con la versione 3 AWS SDK per PHP
- Protezione dell'integrità dei dati con checksum

Client multiregionale Amazon S3 con versione 3 AWS SDK per PHP

La AWS SDK per PHP versione 3 fornisce un client generico multiregionale che può essere utilizzato con qualsiasi servizio. Ciò consente agli utenti di specificare a quale AWS regione inviare un comando fornendo un parametro `@region` di input a qualsiasi comando. Inoltre, l'SDK fornisce un client multiregionale per Amazon S3 che risponde in modo intelligente a errori specifici di Amazon S3 e reindirizza i comandi di conseguenza. Ciò consente agli utenti di usare lo stesso client per comunicare con più regioni. Si tratta di una funzionalità particolarmente utile per gli utenti di Amazon S3 Stream Wrapper con AWS SDK per PHP versione 3, i cui bucket risiedono in più regioni.

Utilizzo di base

Il modello di utilizzo di base di un client Amazon S3 è lo stesso sia che si utilizzi un client S3 standard che la sua controparte multiregionale. L'unica differenza di utilizzo a livello di comando è che una AWS regione può essere specificata utilizzando il parametro di input. `@region`

```
// Create a multi-region S3 client
$s3Client = (new \Aws\Sdk)->createMultiRegionS3(['version' => 'latest']);

// You can also use the client constructor
$s3Client = new \Aws\S3\S3MultiRegionClient([
    'version' => 'latest',
    // Any Region specified while creating the client will be used as the
    // default Region
    'region' => 'us-west-2',
```

```
]);

// Get the contents of a bucket
$objects = $s3Client->listObjects(['Bucket' => $bucketName]);

// If you would like to specify the Region to which to send a command, do so
// by providing an @region parameter
$objects = $s3Client->listObjects([
    'Bucket' => $bucketName,
    '@region' => 'eu-west-1',
]);
```

> **Important**
>
> Quando utilizzi il client Amazon S3 multiregionale, non incontrerai eccezioni di reindirizzamento permanenti. Un client Amazon S3 standard genererà un'istanza di `Aws\S3\Exception\PermanentRedirectException` quando un comando viene inviato alla regione sbagliata. Un client basato su più regioni rispedisce invece il comando alla regione corretta.

Cache Bucket Region

I client multiregionali Amazon S3 mantengono una cache interna delle AWS regioni in cui risiedono determinati bucket. Per impostazione predefinita, ogni client ha la propria cache in memoria. Per condividere una cache tra client o processi, fornisci un'istanza di `Aws\CacheInterface` come opzione `bucket_region_cache` per il client basato su più regioni.

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

Stream wrapper Amazon S3 con versione 3 AWS SDK per PHP

Lo stream wrapper di Amazon S3 consente di archiviare e recuperare dati da Amazon S3 utilizzando funzioni PHP integrate, come`file_get_contents`,,,,, e. `fopen` `copy` `rename` `unlink` `mkdir` `rmdir`

È necessario registrare lo stream wrapper Amazon S3 per utilizzarlo.

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

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

Ciò consente di accedere a bucket e oggetti archiviati in Amazon S3 utilizzando `s3://` il protocollo. Lo stream wrapper di Amazon S3 accetta stringhe che contengono un nome di bucket seguito da una barra e una chiave oggetto o un prefisso opzionale:. `s3://<bucket>[/<key-or-prefix>]`

> **Note**
>
> Il wrapper di flusso è progettato per l'utilizzo con oggetti e bucket su cui si dispone almeno delle autorizzazioni di lettura. Pertanto, il tuo utente dovrebbe disporre delle autorizzazioni necessarie per eseguire `ListBucket` sui bucket e `GetObject` sugli oggetti con cui deve interagire. Per i casi d'uso in cui non disponi di questo livello di autorizzazione, ti consigliamo di utilizzare direttamente le operazioni del client Amazon S3.

Scarica i dati

È possibile acquisire i contenuti di un oggetto utilizzando `file_get_contents`. Tuttavia, questa funzione deve essere utilizzata con cautela, in quanto carica nella memoria tutti i contenuti dell'oggetto.

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

`fopen()`Utilizzalo quando lavori con file di grandi dimensioni o se devi eseguire lo streaming di dati da 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**
>
> Gli errori di scrittura file vengono restituiti solo quando si chiama la funzione `fflush` . Questi errori non vengono restituiti quando si chiama una funzione `fclose` senza svuotamento. Il valore restituito per `fclose` sarà `true` se chiude il flusso, indipendentemente da eventuali errori riportati in risposta alla funzione interna `fflush`. Questi errori non vengono restituiti nemmeno quando si chiama `file_put_contents` per via del modo in cui PHP implementa questa funzione.

Apri stream ricercabili

I flussi aperti in modalità "r" consentono la lettura dei dati solo dal flusso e non supportano la ricerca per impostazione predefinita. In questo modo i dati possono essere scaricati da Amazon S3 in un vero e proprio streaming, in modo che i byte letti in precedenza non debbano essere inseriti nel buffer in memoria. Se è necessario che un flusso supporti la ricerca, è possibile trasferire `seekable` nelle opzioni del contesto del flusso di una funzione.

```
$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);
}
```

L'apertura di flussi che supportano la ricerca consente di cercare byte letti in precedenza. Non è possibile passare a byte che non sono stati ancora letti dal server remoto. Per consentire ai dati letti in precedenza di essere richiamati, viene eseguito il buffering dei dati in un flusso PHP temporaneo utilizzando un decoratore di flussi. Quando la quantità di dati memorizzati nella cache supera i 2 MB, i dati nel flusso temporaneo vengono trasferiti dalla memoria al disco. Tienilo a mente quando scarichi file di grandi dimensioni da Amazon S3 utilizzando l'impostazione del contesto dello `seekable` stream.

Carica dati

Puoi caricare dati su Amazon S3 utilizzando. `file_put_contents()`

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

È possibile caricare file di dimensioni maggiori tramite streaming dei dati utilizzando `fopen()` e una modalità di accesso ai flussi "w", "x" o "a". Lo stream wrapper di Amazon S3 non supporta flussi di lettura e scrittura simultanei (ad esempio «r+», «w+», ecc.). Questo perché il protocollo HTTP non consente la lettura e la scrittura simultanee.

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

> **Note**
>
> Amazon S3 richiede che venga specificata un'intestazione Content-Length prima dell'invio del payload di una richiesta. Pertanto, viene eseguito internamente il buffering dei dati da caricare in un'operazione `PutObject` utilizzando un flusso PHP temporaneo fino a quando il flusso non viene svuotato o chiuso.

> **Note**
>
> Gli errori di scrittura file vengono restituiti solo quando si chiama la funzione `fflush` . Questi errori non vengono restituiti quando si chiama una funzione `fclose` senza svuotamento. Il valore restituito per `fclose` sarà `true` se chiude il flusso, indipendentemente da eventuali errori riportati in risposta alla funzione interna `fflush`. Questi errori non vengono restituiti

nemmeno quando si chiama `file_put_contents` per via del modo in cui PHP implementa questa funzione.

modalità fopen

La funzione fopen() di PHP richiede che venga specificata un'opzione `$mode`. L'opzione della modalità specifica se i dati possono essere letti o scritti in un flusso e se il file deve esistere durante l'apertura di un flusso.

Lo stream wrapper di Amazon S3 supporta le seguenti modalità per gli stream destinati a oggetti Amazon S3.

r	Un flusso di sola lettura in cui l'oggetto deve già esistere.
w	Un flusso di sola scrittura. Se l'oggetto esiste già, viene sovrascritto.
a	Un flusso di sola scrittura. Se l'oggetto esiste già, viene scaricato in uno stream temporane o e le eventuali scritture sullo stream vengono aggiunte ai dati caricati in precedenza.
x	Un flusso di sola scrittura. Se l'oggetto esiste già, viene generato un errore.

Altre funzioni dell'oggetto

Gli stream wrapper consentono a diverse funzioni PHP integrate di funzionare con un sistema personalizzato come Amazon S3. Ecco alcune delle funzioni che lo stream wrapper di Amazon S3 ti consente di eseguire con oggetti archiviati in Amazon S3.

unlink()	Consente di eliminare un oggetto da un bucket.
	`// Delete an object from a bucket`
	`unlink('s3://bucket/key');`

	È possibile trasferire qualsiasi opzione disponibile all'operazione `DeleteObject` per modificare il modo in cui l'oggetto viene eliminato (ad esempio, specificando una versione dell'oggetto specifica).
	`// Delete a specific version of an object from a bucket`
	`unlink('s3://bucket/key', stream_context_create([`
	`'s3' => ['VersionId' => '123']`
	`]);`

filesize()	Consente di ottenere le dimensioni di un oggetto.
	`// Get the Content-Length of an object`
	`$size = filesize('s3://bucket/key', );`

is_file()	Verifica se un URL è un file.
	`if (is_file('s3://bucket/key')) {`
	`echo 'It is a file!';`
	`}`

file_exists ()	Verifica se un oggetto esiste.
	`if (file_exists('s3://bucket/key')) {`
	`echo 'It exists!';`
	`}`

filetype()	Verifica se un URL è associato a un file o a un bucket (directory).

file()	Carica i contenuti di un oggetto in una serie di righe. È possibile trasferire qualsiasi opzione disponibile all'operazione `GetObject` per modificare il modo in cui il file viene scaricato.
filemtime()	Consente di ottenere la data dell'ultima modifica di un oggetto.
rename()	Consente di rinominare un oggetto copiandolo ed eliminando l'oggetto di origine. È possibile trasferire opzioni disponibili per le operazioni `CopyObject` e `DeleteObject` ai parametri del contesto del flusso per modificare il modo in cui l'oggetto viene copiato ed eliminato.

> **Note**
>
> Sebbene `copy` in genere funzioni con lo stream wrapper di Amazon S3, alcuni errori potrebbero non essere segnalati correttamente a causa delle caratteristiche interne della funzione in PHP. `copy` Ti consigliamo invece di utilizzare un'istanza di AwSS3. ObjectCopier

Lavora con bucket e cartelle

`mkdir()`Da utilizzare per lavorare con i bucket

Puoi creare e sfogliare i bucket Amazon S3 in modo simile a come PHP ti consente di creare e attraversare le directory sul tuo file system.

Ecco un esempio che crea un bucket.

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

> **Note**
>
> Nell'aprile 2023, Amazon S3 ha abilitato automaticamente S3 Block Public Access e disabilitato gli elenchi di controllo degli accessi per tutti i bucket appena creati. Questa

modifica influisce anche sul funzionamento della `mkdir` funzione con `StreamWrapper` le autorizzazioni e. ACLs Ulteriori informazioni sono disponibili in questo AWS articolo Cosa c'è di nuovo con.

È possibile passare le opzioni di contesto dello stream al `mkdir()` metodo per modificare il modo in cui il bucket viene creato utilizzando i parametri disponibili per l'CreateBucketoperazione.

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

È possibile eliminare bucket utilizzando la funzione `rmdir()`.

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

> **Note**
>
> Un bucket può essere eliminato solo se è vuoto.

Utilizzatelo **`mkdir()`** per lavorare con le cartelle

Dopo aver creato un bucket, è possibile `mkdir()` utilizzarlo per creare oggetti che funzionano come cartelle come in un file system.

Il seguente frammento di codice aggiunge un oggetto cartella denominato 'my-folder' al bucket esistente denominato 'amzn-s3-demo-bucket'. Utilizzate il carattere forward slash (/) per separare il nome di un oggetto cartella dal nome del bucket e da qualsiasi nome di cartella aggiuntivo.

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

La nota precedente sulle modifiche alle autorizzazioni dopo aprile 2023 entra in gioco anche quando crei oggetti delle cartelle. Questo post del blog contiene informazioni su come modificare le autorizzazioni, se necessario.

Utilizzate la `rmdir()` funzione per eliminare un oggetto cartella vuoto, come illustrato nel seguente frammento.

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

Elenca il contenuto di un bucket

Puoi usare le funzioni PHP opendir (), readdir (), rewinddir () e closedir () con lo stream wrapper Amazon S3 per attraversare il contenuto di un bucket. È possibile passare i parametri disponibili all'operazione come opzioni di contesto di flusso personalizzate alla funzione per modificare la modalità di elenco degli oggetti. ListObjects`opendir()`

```
$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);
}
```

È possibile elencare in modo ricorsivo ogni oggetto e prefisso in un bucket utilizzando PHP. RecursiveDirectoryIterator

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

Un altro modo per elencare i contenuti di un bucket in modo ricorsivo ma con un numero inferiore di richieste HTTP è utilizzare la funzione `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";
}
```

Opzioni di contesto di streaming

È possibile personalizzare il client utilizzato dal wrapper di flusso o la cache utilizzata per memorizzare nella cache informazioni sui bucket e sulle chiavi caricate in precedenza trasferendo le opzioni personalizzate relative al contesto del flusso.

Il wrapper di flusso supporta le seguenti opzioni relative al contesto del flusso per ogni operazione.

`client`

L'oggetto `Aws\AwsClientInterface` da utilizzare per eseguire i comandi.

`cache`

Un'istanza di `Aws\CacheInterface` da utilizzare per memorizzare nella cache le statistiche dei file ottenute in precedenza. Per impostazione predefinita, il wrapper di flusso utilizza una cache LRU nella memoria.

Gestione trasferimenti Amazon S3 con AWS SDK per PHP versione 3

Il gestore di trasferimenti Amazon S3 in AWS SDK per PHP viene utilizzato per caricare intere directory in un bucket Amazon S3 e scaricare interi bucket in una directory locale.

Caricamento di una directory locale su Amazon S3

Per eseguire il trasferimento viene utilizzato l'oggetto `Aws\S3\Transfer`. L'esempio seguente mostra come caricare in modo ricorsivo una directory locale di file in un 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.
$manager = new \Aws\S3\Transfer($client, $source, $dest);

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

In questo esempio, abbiamo creato un client Amazon S3, creato un `Transfer` oggetto ed eseguito il trasferimento in modo sincrono. L'esempio precedente mostra la quantità minima di codice necessaria per eseguire un trasferimento. L'oggetto di trasferimento è in grado di eseguire trasferimenti in modo asincrono e offre varie opzioni di configurazione che possono essere utilizzate per personalizzare i trasferimenti.

Puoi caricare i file locali in una «sottocartella» di un bucket Amazon S3 fornendo un key prefix nell'URI. `s3://` L'esempio seguente mostra come caricare file locali su disco in un bucket `bucket` e come archiviare file con il prefisso della chiave `foo`.

```
$source = '/path/to/source/files';
$dest = 's3://bucket/foo';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Scaricare un bucket Amazon S3

Puoi scaricare in modo ricorsivo un bucket Amazon S3 in una directory locale su disco specificando l'argomento `$source` come URI Amazon S3 (ad esempios3://bucket) e `$dest` l'argomento come percorso di una directory locale.

```
// Where the files will be sourced from.
$source = 's3://bucket';

// Where the files will be transferred to.
$dest = '/path/to/destination/dir';

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

> **Note**
>
> SDK creerà automaticamente tutte le directory necessarie durante il download degli oggetti nel bucket.

Puoi includere un prefisso chiave nell'URI di Amazon S3 dopo il bucket per scaricare solo gli oggetti archiviati in una «pseudo-cartella». L'esempio seguente mostra i download dei soli file memorizzati con il prefisso della chiave "/foo" di un determinato bucket.

```
$source = 's3://bucket/foo';
$dest = '/path/to/destination/dir';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
$manager->transfer();
```

Configurazione

Il costruttore dell'oggetto `Transfer` accetta i seguenti argomenti.

`$client`

L'oggetto `Aws\ClientInterface` da utilizzare per eseguire i trasferimenti.

`$source`(**`Iterator`**stringa |)

I dati di origine in fase di trasferimento. Questo può indicare un percorso locale su disco (ad esempio`/path/to/files`) o un bucket Amazon S3 (ad esempio). `s3://bucket` L'URI `s3://` può contenere anche una chiave del prefisso che può essere utilizzata solo per il trasferimento di oggetti con un prefisso comune.

Se l'`$source`argomento è un URI Amazon S3, l'`$dest`argomento deve essere una directory locale (e viceversa).

Oltre a fornire un valore della stringa, è anche possibile fornire un oggetto `\Iterator` che produce nomi di file assoluti. Se si fornisce un'iterazione, è necessario fornire un'opzione `base_dir` nell'associativo `$options`.

`$dest`

La destinazione in cui i file verranno trasferiti. Se l'`$source`argomento è un percorso locale su disco, `$dest` deve essere un URI del bucket Amazon S3 (ad esempio). `s3://bucket` Se l'`$source`argomento è un URI del bucket Amazon S3, l'`$dest`argomento deve essere un percorso locale su disco.

`$options`

Un array associativo delle opzioni di trasferimento. Le seguenti opzioni di trasferimento sono valide:

`add_content_md5` (bool)

Imposta su `true` per calcolare il MD5 checksum per i caricamenti.

`base_dir` (Stringa)

Directory di base dell'origine, se `$source` è un'iterazione. Se l'opzione `$source` non è un array, questa opzione viene ignorata.

`before` (callable)

Un callback da invocare prima di ogni trasferimento. Il callback deve disporre di una firma della funzione, come `function (Aws\Command $command) {...}`. Il comando fornito sarà `GetObject`, `PutObject`, `CreateMultipartUpload`, `UploadPart` o `CompleteMultipartUpload`.

`mup_threshold` (int)

Le dimensioni in byte di un caricamento in più parti da utilizzare al posto di `PutObject`. Il valore predefinito è `16777216` (16 MB).

`concurrency` (int, predefinito=5)

Numero di file da caricare simultaneamente. Il valore ottimale varia in base al numero di file caricati e alle dimensioni medie di ciascun file. Di solito, i file di dimensioni minori possono contare su un valore di simultaneità maggiore rispetto ai file di dimensioni più grandi.

`debug` (bool)

Imposta su `true` per stampare le informazioni di debug per i trasferimenti. Imposta su una risorsa `fopen()` per la scrittura su un flusso specifico invece che su STDOUT.

Trasferimenti asincroni

L'oggetto `Transfer` è un'istanza di `GuzzleHttp\Promise\PromisorInterface`. Pertanto, il trasferimento può verificarsi in modo asincrono e viene avviato chiamando il metodo `promise` dell'oggetto.

```
$source = '/path/to/source/files';
$dest = 's3://bucket';
$manager = new \Aws\S3\Transfer($client, $source, $dest);
```

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

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

La promessa verrà rifiutata se il trasferimento di uno dei file non riesce. È possibile gestire il trasferimento non riuscito in modo asincrono utilizzando il metodo `otherwise` della promessa. La funzione `otherwise` accetta un callback da invocare quando si verifica un errore. Il callback accetta la `$reason` del rifiuto, che, di solito, è un'istanza di `Aws\Exception\AwsException` (anche se un valore di qualsiasi tipo può essere distribuito al callback).

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

Poiché l'oggetto `Transfer` restituisce una promessa, questi trasferimenti possono verificarsi simultaneamente con altre promesse asincrone.

Personalizzazione dei comandi del gestore dei trasferimenti

È possibile impostare opzioni personalizzate per le operazioni eseguite dalla gestione trasferimenti tramite un callback trasmesso al relativo costruttore.

```
$uploader = new Transfer($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';
        }
    },
]);
```

Crittografia lato client di Amazon S3 con la versione 3 AWS SDK per PHP

Con la crittografia lato client, i dati vengono crittografati e decrittografati direttamente nel tuo ambiente. Ciò significa che questi dati vengono crittografati prima di essere trasferiti su Amazon S3 e non ti affidi a un servizio esterno per gestire la crittografia per te. Per le nuove implementazioni, suggeriamo di utilizzare `S3EncryptionClientV2` e sostituire la versione `S3EncryptionMultipartUploaderV2` obsoleta e. `S3EncryptionClient` `S3EncryptionMultipartUploader` È consigliabile che le implementazioni precedenti che utilizzano ancora le versioni obsolete tentino di migrare. `S3EncryptionClientV2`mantiene il supporto per la decrittografia dei dati crittografati utilizzando la versione precedente. `S3EncryptionClient`

AWS SDK per PHP Implementa la crittografia delle buste e utilizza OpenSSL per la crittografia e la decrittografia. L'implementazione è interoperabile con altre che supportano le sue funzionalità. SDKs Inoltre, è compatibile con il flusso di lavoro asincrono basato su promessa dell'SDK.

Guida alla migrazione

Per coloro che stanno cercando di migrare dai client obsoleti ai nuovi client, è disponibile una guida alla migrazione che può essere trovata qui.

Configurazione

Per iniziare a utilizzare la crittografia lato client, è necessario:

- Una chiave di crittografia AWS KMS
- Un bucket S3

Prima di eseguire qualsiasi codice di esempio, configura le tue AWS credenziali. Vedi Credenziali per la AWS SDK per PHP versione 3.

Crittografia

Il caricamento di un oggetto crittografato `S3EncryptionClientV2` richiede tre parametri aggiuntivi oltre ai parametri standard`PutObject`:

- `'@KmsEncryptionContext'`è una coppia chiave-valore che può essere utilizzata per aggiungere un ulteriore livello di sicurezza all'oggetto crittografato. Il client di crittografia deve inserire la stessa chiave, operazione che eseguirà automaticamente durante una chiamata get. Se non desideri alcun contesto aggiuntivo, passa un array vuoto.

- `@CipherOptions`sono configurazioni aggiuntive per la crittografia, tra cui il codice da utilizzare e la dimensione della chiave.
- `@MaterialsProvider`è un provider che gestisce la generazione di una chiave di cifratura e di un vettore di inizializzazione, nonché la crittografia della chiave di cifratura.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV2;

 // Let's construct our S3EncryptionClient using an S3Client
 $encryptionClient = new S3EncryptionClientV2(
     new S3Client([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ])
 );

 $kmsKeyId = 'kms-key-id';
 $materialsProvider = new KmsMaterialsProviderV2(
     new KmsClient([
         'profile' => 'default',
         'region' => 'us-east-1',
         'version' => 'latest',
     ]),
     $kmsKeyId
 );

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

 $result = $encryptionClient->putObject([
     '@MaterialsProvider' => $materialsProvider,
     '@CipherOptions' => $cipherOptions,
     '@KmsEncryptionContext' => ['context-key' => 'context-value'],
     'Bucket' => $bucket,
```

```
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

> **Note**
>
> Oltre agli errori di Amazon S3 e dei servizi AWS KMS basati su Amazon, potresti ricevere `InvalidArgumentException` oggetti generati se non `'@CipherOptions'` sei configurato correttamente.

Decrittografia

Il download e la decrittografia di un oggetto hanno quattro parametri aggiuntivi, due dei quali obbligatori, oltre ai parametri standard. `GetObject` Il client rileverà automaticamente le opzioni di crittografia di base.

- **`'@SecurityProfile'`**: Se impostato su 'V2', solo gli oggetti crittografati in versione compatibile con V2

  il formato può essere decifrato. L'impostazione di questo parametro su 'V2_AND_LEGACY' consente inoltre di decrittografare gli oggetti crittografati in un formato compatibile con V1. Per supportare la migrazione, imposta @ su 'V2_AND_LEGACY'. SecurityProfile Usa 'V2' solo per lo sviluppo di nuove applicazioni.
- **`'@MaterialsProvider'`**è un provider che gestisce la generazione di una chiave di cifratura e di un vettore di inizializzazione, come

  oltre a crittografare la chiave di crittografia.
- **`'@KmsAllowDecryptWithAnyCmk'`**: (opzionale) L'impostazione di questo parametro su true abilita la decrittografia

  senza fornire un id di chiave KMS al costruttore di. MaterialsProvider Il valore predefinito è false.
- **`'@CipherOptions'`**(opzionale) sono configurazioni aggiuntive per la crittografia, tra cui

  cifrario da usare e dimensione della chiave.

```
$result = $encryptionClient->getObject([
```

```
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

> **Note**
>
> Oltre agli errori di Amazon S3 e dei servizi AWS KMS basati su Amazon, potresti ricevere `InvalidArgumentException` oggetti generati se non `'@CipherOptions'` sei configurato correttamente.

Configurazione Cipher

`'Cipher'` (Stringa)

Modalità di cifratura che il client di crittografia utilizza durante la crittografia. Al momento è supportato solo 'gcm'.

> **Important**
>
> PHP è aggiornato nella versione 7.1 per includere i parametri aggiuntivi necessari per crittografare e decrittografare l'utilizzo di OpenSSL per la crittografia GCM. Per le versioni PHP 7.0 e precedenti, un polyfill per il supporto GCM viene fornito e utilizzato dai client di crittografia e. `S3EncryptionClientV2` `S3EncryptionMultipartUploaderV2` Tuttavia, le prestazioni per input di grandi dimensioni saranno molto più lente utilizzando il polyfill rispetto all'implementazione nativa per PHP 7.1+, quindi potrebbe essere necessario aggiornare gli ambienti con versioni PHP precedenti per utilizzarli in modo efficace.

`'KeySize'` (int)

La durata della chiave di crittografia del contenuto da generare per la crittografia. Il valore di default è 256 bit. Le opzioni di configurazione valide sono 256 e 128 bit.

`'Aad'` (Stringa)

"Dati di autenticazione aggiuntivi" facoltativi da includere con il tuo payload crittografato. Queste informazioni sono convalidate per la decrittografia. `Aad` è disponibile solo quando si usa la cifratura "gcm".

> **Important**
>
> I dati di autenticazione aggiuntivi non sono supportati da tutti AWS SDKs e pertanto altri SDKs potrebbero non essere in grado di decrittografare i file crittografati utilizzando questo parametro.

Strategie relative ai metadati

Puoi anche fornire un'istanza di una classe che implementa `Aws\Crypto\MetadataStrategyInterface`. Questa semplice interfaccia gestisce il salvataggio e il caricamento di `Aws\Crypto\MetadataEnvelope` che contiene i materiali per la crittografia della busta. L'SDK fornisce due classi che implementano questo: `Aws\S3\Crypto\HeadersMetadataStrategy` e `Aws\S3\Crypto\InstructionFileMetadataStrategy`. `HeadersMetadataStrategy` viene utilizzato per impostazione predefinita.

```
$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

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

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V2',
```

```
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Le costanti per il nome della classe `HeadersMetadataStrategy` e `InstructionFileMetadataStrategy` possono anche essere fornite chiamando ::class.

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

> **Note**
>
> Se si verifica un errore dopo che un file di istruzioni è stato caricato, non verrà automaticamente eliminato.

Caricamenti in più parti

Anche l'esecuzione di un caricamento in più parti con crittografia lato client è possibile. `Aws\S3\Crypto\S3EncryptionMultipartUploaderV2`Prepara il flusso sorgente per la crittografia prima del caricamento. La creazione di uno è simile a utilizzare `Aws\S3\MultipartUploader` e `Aws\S3\Crypto\S3EncryptionClientV2`. `S3EncryptionMultipartUploaderV2` può gestire la stessa opzione `'@MetadataStrategy'` come `S3EncryptionClientV2`, oltre a tutte le configurazioni `'@CipherOptions'` disponibili.

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
```

```
    $kmsKeyId
);

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

$multipartUploader = new S3EncryptionMultipartUploaderV2(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
    [
        '@MaterialsProvider' => $materialsProvider,
        '@CipherOptions' => $cipherOptions,
        'bucket' => $bucket,
        'key' => $key,
    ]
);
$multipartUploader->upload();
```

> **Note**
>
> Oltre agli errori di Amazon S3 e dei servizi AWS KMS basati su Amazon, potresti ricevere `InvalidArgumentException` oggetti generati se non `'@CipherOptions'` sei configurato correttamente.

Protezione dell'integrità dei dati con checksum

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) offre la possibilità di specificare un checksum quando carichi un oggetto. Quando specifichi un checksum, questo viene memorizzato con l'oggetto e può essere convalidato quando l'oggetto viene scaricato.

I checksum forniscono un ulteriore livello di integrità dei dati durante il trasferimento dei file. Con i checksum, è possibile verificare la coerenza dei dati confermando che il file ricevuto corrisponde al

file originale. Per ulteriori informazioni sui checksum con Amazon S3, consulta la Guida per l'utente di Amazon Simple Storage Service, che include gli algoritmi supportati.

Hai la flessibilità di scegliere l'algoritmo più adatto alle tue esigenze e lasciare che sia l'SDK a calcolare il checksum. In alternativa, puoi fornire un valore di checksum precalcolato utilizzando uno degli algoritmi supportati.

> **Note**
>
> L'SDK fornisce anche impostazioni globali per la protezione dell'integrità dei dati che puoi impostare esternamente, come puoi leggere nella Guida di riferimento agli strumenti.AWS SDKs

Discutiamo i checksum in due fasi di richiesta: caricamento di un oggetto e download di un oggetto.

Caricamento di un oggetto

Se non fornisci un algoritmo di checksum con la richiesta, il comportamento del checksum varia a seconda della versione dell'SDK che usi, come mostrato nella tabella seguente.

Comportamento del checksum quando non viene fornito alcun algoritmo di checksum

Utilizza un valore di checksum precalcolato

Un valore di checksum precalcolato fornito con la richiesta disabilita il calcolo automatico da parte dell'SDK e utilizza invece il valore fornito.

L'esempio seguente mostra una richiesta con un checksum precalcolato. SHA256

Se Amazon S3 determina che il valore del checksum non è corretto per l'algoritmo specificato, il servizio restituisce una risposta di errore.

Caricamenti in più parti

Puoi anche utilizzare i checksum con caricamenti in più parti.

Donwload di un oggetto

La richiesta nel seguente frammento indica all'SDK di convalidare il checksum nella risposta calcolando il checksum e confrontando i valori.

> **Note**
>
> Se l'oggetto non è stato caricato con un checksum, non viene effettuata alcuna convalida.

Esempi di codice con linee guida per AWS SDK per PHP

Questa sezione contiene esempi di codice che mostrano AWS scenari comuni che utilizzano AWS SDK per PHP.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Argomenti

- CloudFront Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3
- Firma di richieste di CloudSearch dominio Amazon personalizzate con AWS SDK per PHP la versione 3
- CloudWatch Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3
- EC2 Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3
- Firma di una richiesta OpenSearch di ricerca Amazon Service con AWS SDK per PHP la versione 3
- AWS Identity and Access Management esempi utilizzando la AWS SDK per PHP versione 3
- AWS Key Management Service esempi utilizzando la AWS SDK per PHP versione 3
- Esempi di Amazon Kinesis con la versione 3 AWS SDK per PHP
- AWS Elemental MediaConvert esempi utilizzando la AWS SDK per PHP versione 3
- Esempi di Amazon S3 con la AWS SDK per PHP versione 3
- Gestione dei segreti utilizzando l'API Secrets Manager e la AWS SDK per PHP versione 3
- Esempi di Amazon SES con la AWS SDK per PHP versione 3
- Esempi di Amazon SNS con la versione 3 AWS SDK per PHP
- Esempi di Amazon SQS con la AWS SDK per PHP versione 3
- Invia eventi agli endpoint EventBridge globali di Amazon

CloudFront Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3

Amazon CloudFront è un servizio AWS Web che velocizza la pubblicazione di contenuti Web statici e dinamici dal tuo server Web o da un AWS server, come Amazon S3. CloudFront fornisce contenuti attraverso una rete mondiale di data center denominati edge location. Quando un utente richiede i contenuti con cui li distribuisci CloudFront, viene indirizzato all'edge location che offre la latenza più bassa. Se i contenuti non sono già archiviati nella cache, CloudFront recupera una copia dal server di origine, la serve e quindi la memorizza nella cache per le richieste future.

Per ulteriori informazioni CloudFront, consulta l'Amazon CloudFront Developer Guide.

Tutto il codice di esempio per la AWS SDK per PHP versione 3 è disponibile qui GitHub.

Gestione delle CloudFront distribuzioni Amazon utilizzando l' CloudFront API e la AWS SDK per PHP versione 3

Amazon CloudFront memorizza nella cache i contenuti nelle edge location di tutto il mondo per accelerare la distribuzione di file statici e dinamici archiviati sul tuo server o su un servizio Amazon come Amazon S3 e Amazon. EC2 Quando gli utenti richiedono contenuti dal tuo sito Web, li CloudFront invia dalla edge location più vicina, se il file è memorizzato nella cache. Altrimenti CloudFront recupera una copia del file, lo serve e quindi lo memorizza nella cache per la richiesta successiva. Il caching dei contenuti in una edge location riduce la latenza di richieste utente simili in tale area.

Per ogni CloudFront distribuzione creata, specificate dove si trova il contenuto e come distribuirlo quando gli utenti effettuano richieste. Questo argomento descrive le distribuzioni di file statici e dinamici, ad esempio HTML, CSS, JSON e file di immagine. Per informazioni sull'utilizzo CloudFront con video on demand, consulta Video on demand e live streaming con CloudFront.

Gli esempi seguenti mostrano come:

- Crea una distribuzione utilizzando CreateDistribution.
- Ottieni una distribuzione usando GetDistribution.
- Elenca le distribuzioni utilizzando ListDistributions.
- Aggiorna le distribuzioni utilizzando. UpdateDistributions
- Disabilita le distribuzioni utilizzando. DisableDistribution
- Elimina le distribuzioni utilizzando. DeleteDistributions

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon CloudFront, consulta l'Amazon CloudFront Developer Guide.

Crea una CloudFront distribuzione

Crea una distribuzione da un bucket Amazon S3. Nell'esempio seguente, i parametri opzionali vengono commentati, ma i valori di default vengono visualizzati. Per aggiungere personalizzazioni alla distribuzione, rimuovi i commenti dal valore e dal parametro all'interno di `$distribution`.

Per creare una CloudFront distribuzione, usa l'CreateDistributionoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);

        $message = '';

        if (isset($result['Distribution']['Id'])) {
            $message = 'Distribution created with the ID of ' .
                $result['Distribution']['Id'];
        }

        $message .= ' and an effective URI of ' .
            $result['@metadata']['effectiveUri'] . '.';
```

```
        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'my-bucket-name.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
    $defaultCacheBehavior = [
        'AllowedMethods' => [
            'CachedMethods' => [
                'Items' => ['HEAD', 'GET'],
                'Quantity' => 2
            ],
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Compress' => false,
        'DefaultTTL' => 0,
        'FieldLevelEncryptionId' => '',
        'ForwardedValues' => [
            'Cookies' => [
                'Forward' => 'none'
            ],
            'Headers' => [
                'Quantity' => 0
            ],
            'QueryString' => false,
            'QueryStringCacheKeys' => [
                'Quantity' => 0
            ]
        ],
        'LambdaFunctionAssociations' => ['Quantity' => 0],
        'MaxTTL' => 0,
        'MinTTL' => 0,
        'SmoothStreaming' => false,
        'TargetOriginId' => $originName,
        'TrustedSigners' => [
            'Enabled' => false,
```

```
            'Quantity' => 0
        ],
        'ViewerProtocolPolicy' => 'allow-all'
    ];
    $enabled = false;
    $origin = [
        'Items' => [
            [
                'DomainName' => $s3BucketURL,
                'Id' => $originName,
                'OriginPath' => '',
                'CustomHeaders' => ['Quantity' => 0],
                'S3OriginConfig' => ['OriginAccessIdentity' => '']
            ]
        ],
        'Quantity' => 1
    ];
    $distribution = [
        'CallerReference' => $callerReference,
        'Comment' => $comment,
        'DefaultCacheBehavior' => $defaultCacheBehavior,
        'Enabled' => $enabled,
        'Origins' => $origin
    ];

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.
// createsTheS3Distribution();
```

Recupera una distribuzione CloudFront

Per recuperare lo stato e i dettagli di una CloudFront distribuzione specificata, utilizzate l'GetDistributionoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Elenca CloudFront le distribuzioni

Ottieni un elenco delle CloudFront distribuzioni esistenti nella AWS regione specificata dal tuo account corrente utilizzando l'ListDistributionsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Aggiorna una distribuzione CloudFront

L'aggiornamento di una CloudFront distribuzione è simile alla creazione di una distribuzione. Tuttavia, quando aggiorni una distribuzione, sono richiesti più campi e devono essere inclusi tutti i valori. Per apportare modifiche a una distribuzione esistente, ti consigliamo di recuperare la distribuzione esistente e aggiornare i valori da modificare nell'array `$distribution`.

Per aggiornare una CloudFront distribuzione specificata, usa l'UpdateDistributionoperazione.

Importazioni

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

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

Codice di esempio

```
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();
```

Disabilita una CloudFront distribuzione

Per disattivare o rimuovere una distribuzione, modifica il relativo stato da distribuito a disabilitato.

Per disabilitare la CloudFront distribuzione specificata, utilizzare l'DisableDistributionoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Eliminare una CloudFront distribuzione

Una volta che una distribuzione è in uno stato disabilitato, puoi eliminarla.

Per rimuovere una CloudFront distribuzione specificata, utilizzare l'DeleteDistributionoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Gestione delle CloudFront invalidazioni di Amazon utilizzando l' CloudFront API e la versione 3 AWS SDK per PHP

Amazon CloudFront memorizza nella cache copie di file statici e dinamici in edge location in tutto il mondo. Per rimuovere o aggiornare un file su tutte le edge location, crea un invalidamento per ogni file o per un gruppo di file.

Ogni mese di calendario, i primi 1.000 invalidamenti sono gratuiti. Per ulteriori informazioni sulla rimozione di contenuti da una CloudFront edge location, consulta Invalidating Files.

Gli esempi seguenti mostrano come:

- Crea un'invalidazione della distribuzione utilizzando. CreateInvalidation
- Ottieni un'invalidazione della distribuzione utilizzando. GetInvalidation
- Elenca le distribuzioni utilizzando. ListInvalidations

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon CloudFront, consulta l'Amazon CloudFront Developer Guide.

Crea un'invalidazione della distribuzione

Crea un'invalidazione della CloudFront distribuzione specificando la posizione del percorso per i file che devi rimuovere. Questo esempio consente di invalidare tutti i file nella distribuzione, ma puoi identificare file specifici in `Items`.

Per creare un'invalidazione CloudFront della distribuzione, usa l'operazione. CreateInvalidation

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Ottieni un'invalidazione della distribuzione

Per recuperare lo stato e i dettagli sull'invalidazione di una CloudFront distribuzione, usa l'operazione. GetInvalidation

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Elenca le invalidazioni della distribuzione

Per elencare tutte le invalidazioni correnti CloudFront della distribuzione, usa l'operazione. ListInvalidations

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
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();
```

Firmare Amazon CloudFront URLs con AWS SDK per PHP la versione 3

Signed ti URLs consente di fornire agli utenti l'accesso ai tuoi contenuti privati. Un URL firmato include ulteriori informazioni (ad esempio, un'ora di scadenza) che offrono un maggiore controllo sull'accesso ai contenuti. Queste informazioni aggiuntive appaiono in una dichiarazione di policy, basata su una policy predefinita o personalizzata. Per informazioni su come configurare le distribuzioni private e sul motivo per cui devi firmare URLs, consulta Serving Private Content through Amazon CloudFront nella Amazon CloudFront Developer Guide.

- Crea un CloudFront URL Amazon firmato utilizzando getSigneDurl.
- Crea un CloudFront cookie Amazon firmato utilizzando getSignedCookie.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon CloudFront, consulta l'Amazon CloudFront Developer Guide.

Firma CloudFront URLs per distribuzioni private

Puoi firmare un URL utilizzando il CloudFront client nell'SDK. Innanzitutto, è necessario creare un oggetto `CloudFrontClient`. Puoi firmare un CloudFront URL per una risorsa video utilizzando una politica predefinita o personalizzata.

Importazioni

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

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

Codice di esempio

```
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();
```

Utilizza una politica personalizzata durante la creazione CloudFront URLs

Per utilizzare una policy personalizzata, offri la chiave `policy` anziché `expires`.

Importazioni

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

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

Codice di esempio

```
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();
```

Usa un URL CloudFront firmato

Il formato dell'URL firmato varia a seconda che l'URL che si sta firmando utilizzi lo schema "HTTP" o "RTMP". Nel caso di "HTTP", viene restituito un URL completo e assoluto. Se viene utilizzato lo schema "RTMP", per praticità viene restituito solo un URL relativo. Ciò è dovuto al fatto che alcuni lettori richiedono che l'host e il percorso vengano forniti come parametri separati.

L'esempio seguente mostra come utilizzare l'URL firmato per creare una pagina Web che visualizza un video utilizzando. JWPlayer Lo stesso tipo di tecnica si applicherebbe ad altri giocatori, ad esempio FlowPlayer, ma richiederebbe un codice lato client diverso.

```
<html>
<head>
    <title>|CFlong| Streaming Example</title>
    <script type="text/javascript" src="https://example.com/jwplayer.js"></script>
</head>
<body>
    <div id="video">The canned policy video will be here.</div>
    <script type="text/javascript">
        jwplayer('video').setup({
            file: "<?= $streamHostUrl ?>/cfx/st/<?= $signedUrlCannedPolicy ?>",
            width: "720",
            height: "480"
        });
    </script>
</body>
</html>
```

CloudFront Cookie di firma per distribuzioni private

In alternativa a quelli firmati URLs, puoi anche concedere ai clienti l'accesso a una distribuzione privata tramite cookie firmati. I cookie firmati consentono di fornire accesso a più file con restrizioni, ad esempio, tutti i file per un video in formato HLS o tutti i file nell'area abbonati di un sito Web. Per ulteriori informazioni sul motivo per cui potresti voler utilizzare cookie firmati anziché firmati URLs (o viceversa), consulta Scelta tra cookie firmati URLs e firmati nella Amazon CloudFront Developer Guide.

Il processo di creazione di un cookie firmato è simile a quello di creazione di un URL firmato. L'unica differenza è il metodo che viene chiamato (`getSignedCookie` anziché `getSignedUrl`).

Importazioni

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

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

Codice di esempio

```
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();
```

Utilizza una politica personalizzata per la creazione CloudFront dei cookie

Analogamente a `getSignedUrl`, è possibile fornire un parametro `'policy'` anziché un parametro `expires` e un parametro `url` per firmare un cookie con una policy personalizzata. Una policy personalizzata può contenere caratteri jolly nella chiave risorsa. In questo modo è possibile creare un singolo cookie firmato per più file.

`getSignedCookie` restituisce una serie di coppie chiave-valore che devono essere impostate come cookie per concedere l'accesso a una distribuzione privata.

Importazioni

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

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

Codice di esempio

```
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();
```

Invia CloudFront cookie al client Guzzle

È anche possibile trasferire questi cookie a una funzione `GuzzleHttp\Cookie\CookieJar` per l'utilizzo con un client Guzzle.

```
use GuzzleHttp\Client;
use GuzzleHttp\Cookie\CookieJar;

$distribution = "example-distribution.cloudfront.net";
$client = new \GuzzleHttp\Client([
    'base_uri' => "https://$distribution",
    'cookies' => CookieJar::fromArray($signedCookieCustomPolicy, $distribution),
]);

$client->get('video.mp4');
```

Per ulteriori informazioni, consulta Using Signed Cookies nella Amazon CloudFront Developer Guide.

Firma di richieste di CloudSearch dominio Amazon personalizzate con AWS SDK per PHP la versione 3

Le richieste di CloudSearch dominio Amazon possono essere personalizzate oltre a quanto supportato da AWS SDK per PHP. Nei casi in cui è necessario effettuare richieste personalizzate a domini protetti dall'autenticazione IAM, è possibile utilizzare i fornitori di credenziali e i firmatari dell'SDK per firmare qualsiasi richiesta PSR-7.

Ad esempio, se segui la Guida alle operazioni di base di Cloud Search e desideri utilizzare un dominio protetto di IAM per la Fase 3, devi firmare ed eseguire la tua richiesta come segue.

Gli esempi seguenti mostrano come:

- Firma una richiesta con il protocollo di firma utilizzando SignatureV4. AWS

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Firma la richiesta di CloudSearch dominio Amazon

Importazioni

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

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

Codice di esempio

```
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 Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3

Amazon CloudWatch (CloudWatch) è un servizio web che monitora le tue risorse Amazon Web Services e le applicazioni su cui esegui AWS in tempo reale. Puoi utilizzarlo CloudWatch per raccogliere e tenere traccia delle metriche, che sono variabili che puoi misurare per le tue risorse e applicazioni. CloudWatch gli allarmi inviano notifiche o apportano automaticamente modifiche alle risorse che stai monitorando in base a regole da te definite.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Argomenti

- Utilizzo degli CloudWatch allarmi Amazon con la AWS SDK per PHP versione 3
- Ottenere metriche da Amazon CloudWatch con la AWS SDK per PHP versione 3
- Pubblicazione di metriche personalizzate in Amazon CloudWatch con la AWS SDK per PHP versione 3

- Invio di eventi ad Amazon CloudWatch Events con AWS SDK per PHP la versione 3
- Utilizzo delle azioni di allarme con gli CloudWatch allarmi di Amazon con la AWS SDK per PHP versione 3

Utilizzo degli CloudWatch allarmi Amazon con la AWS SDK per PHP versione 3

Una CloudWatch sveglia Amazon rileva una singola metrica per un periodo di tempo specificato. L'allarme esegue una o più operazioni basate sul valore del parametro relativo a una soglia prestabilita per un certo numero di periodi.

Gli esempi seguenti mostrano come:

- Descrivi un allarme utilizzando DescribeAlarms.
- Crea un allarme usando PutMetricAlarm.
- Eliminare un allarme utilizzando DeleteAlarms.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Descrizione di allarmi

Importazioni

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

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

Codice di esempio

```
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();
```

Creazione di un allarme

Importazioni

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

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

Codice di esempio

```
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();
```

Eliminare allarmi

Importazioni

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

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

Codice di esempio

```
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();
```

Ottenere metriche da Amazon CloudWatch con la AWS SDK per PHP versione 3

I parametri sono dati riguardanti le prestazioni dei sistemi. Puoi abilitare il monitoraggio dettagliato di alcune risorse, come le EC2 istanze Amazon, o dei parametri delle tue applicazioni.

Gli esempi seguenti mostrano come:

- Elenca le metriche utilizzando. ListMetrics
- Recupera gli allarmi per una metrica utilizzando. DescribeAlarmsForMetric
- Ottieni statistiche per una metrica specificata utilizzando. GetMetricStatistics

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Elencare parametri

Importazioni

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

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

Codice di esempio

```
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();
```

Recupera gli allarmi per una metrica

Importazioni

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

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

Codice di esempio

```
function describeAlarmsForMetric(
    $cloudWatchClient,
    $metricName,
    $namespace,
    $dimensions
) {
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,
            'Namespace' => $namespace,
            'Dimensions' => $dimensions
        ]);

        $message = '';

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= 'At the effective URI of ' .
                $result['@metadata']['effectiveUri'] . ":\n\n";

            if (
                (isset($result['MetricAlarms'])) and
                (count($result['MetricAlarms']) > 0)
            ) {
                $message .= 'Matching alarms for ' . $metricName . ":\n\n";

                foreach ($result['MetricAlarms'] as $alarm) {
                    $message .= $alarm['AlarmName'] . "\n";
                }
            } else {
                $message .= 'No matching alarms found for ' . $metricName . '.';
            }
        } else {
            $message .= 'No matching alarms found for ' . $metricName . '.';
        }

        return $message;
    } catch (AwsException $e) {
```

```
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value' => 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'my-bucket'
        ]
    ];

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo describeAlarmsForMetric(
        $cloudWatchClient,
        $metricName,
        $namespace,
        $dimensions
    );
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

Ottenere le statistiche di un parametro

Importazioni

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

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

Codice di esempio

```
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' => 'my-bucket'
        ]
    ];
    $startTime = strtotime('-3 days');
    $endTime = strtotime('now');
    $period = 86400; // Seconds. (1 day = 86400 seconds.)
    $statistics = array('Average');
    $unit = 'Bytes';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
```

```
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
    */
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

Pubblicazione di metriche personalizzate in Amazon CloudWatch con la AWS SDK per PHP versione 3

I parametri sono dati riguardanti le prestazioni dei sistemi. Un allarme monitora un singolo parametro per un periodo di tempo specificato. L'allarme esegue una o più operazioni basate sul valore del parametro relativo a una soglia prestabilita per un certo numero di periodi.

Gli esempi seguenti mostrano come:

- Pubblica dati metrici utilizzando. PutMetricData
- Crea un allarme utilizzando PutMetricAlarm.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Pubblica i dati metrici

Importazioni

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

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

Codice di esempio

```
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();
```

Creazione di un allarme

Importazioni

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

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

Codice di esempio

```
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();
```

Invio di eventi ad Amazon CloudWatch Events con AWS SDK per PHP la versione 3

CloudWatch Events fornisce un flusso quasi in tempo reale di eventi di sistema che descrivono le modifiche nelle risorse di Amazon Web Services (AWS) a qualsiasi destinazione. Utilizzando semplici regole, puoi abbinare gli eventi e instradarli verso una o più funzioni o flussi target.

Gli esempi seguenti mostrano come:

- Crea una regola utilizzando PutRule.
- Aggiungi obiettivi a una regola utilizzando PutTargets.
- Invia eventi personalizzati a CloudWatch Events utilizzando PutEvents.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creazione di una regola

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

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

Aggiungere obiettivi a una regola

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

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

Invio di eventi personalizzati

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

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

Utilizzo delle azioni di allarme con gli CloudWatch allarmi di Amazon con la AWS SDK per PHP versione 3

Usa le azioni di allarme per creare allarmi che interrompono, terminano, riavviano o ripristinano automaticamente le tue istanze Amazon. EC2 Puoi utilizzare le operazioni di arresto o termine quando non è più necessaria l'esecuzione di un'istanza. È possibile utilizzare le azioni di riavvio e ripristino per riavviare automaticamente tali istanze.

Gli esempi seguenti mostrano come:

- Abilita azioni per allarmi specifici utilizzando. EnableAlarmActions
- Disabilita le azioni per allarmi specifici utilizzando. DisableAlarmActions

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Attivare le operazioni di allarme

Importazioni

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

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

Codice di esempio

```
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();
```

Disattivare le operazioni di allarme

Importazioni

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

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

Codice di esempio

```
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 Esempi di Amazon che utilizzano la AWS SDK per PHP versione 3

Amazon Elastic Compute Cloud (Amazon EC2) è un servizio Web che fornisce hosting di server virtuali nel cloud. È progettato per semplificare il cloud computing su scala web per gli sviluppatori fornendo una capacità di elaborazione ridimensionabile.

Tutto il codice di esempio per il è disponibile qui. AWS SDK per PHP GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Argomenti

- Gestione delle EC2 istanze Amazon utilizzando la AWS SDK per PHP versione 3
- Utilizzo di indirizzi IP elastici con Amazon EC2 con AWS SDK per PHP versione 3
- Utilizzo di regioni e zone di disponibilità per Amazon EC2 con AWS SDK per PHP la versione 3
- Utilizzo delle coppie di EC2 chiavi Amazon con AWS SDK per PHP la versione 3
- Lavorare con i gruppi di sicurezza in Amazon EC2 con AWS SDK per PHP la versione 3

Gestione delle EC2 istanze Amazon utilizzando la AWS SDK per PHP versione 3

Gli esempi seguenti mostrano come:

- Descrivi EC2 le istanze Amazon utilizzando DescribeInstances.
- Abilita il monitoraggio dettagliato per un'istanza in esecuzione utilizzando MonitorInstances.
- Disabilita il monitoraggio per un'istanza in esecuzione utilizzando UnmonitorInstances.
- Avvia un'AMI supportata da Amazon EBS che hai interrotto in precedenza, utilizzando. StartInstances
- Interrompi l'utilizzo di un'istanza supportata da Amazon EBS. StopInstances
- Richiedi il riavvio di una o più istanze utilizzando. RebootInstances

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Descrivere le istanze

Importazioni

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

use Aws\Ec2\Ec2Client;
```

Codice di esempio

```
$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";
    }
}
```

Attivare e disattivare il monitoraggio

Importazioni

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

Codice di esempio

```
$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);
```

Avviare e arrestare un'istanza

Importazioni

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

Codice di esempio

```
$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);
```

Riavviare un'istanza

Importazioni

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

Codice di esempio

```
$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);
```

Utilizzo di indirizzi IP elastici con Amazon EC2 con AWS SDK per PHP versione 3

Un indirizzo IP elastico è un indirizzo IP statico progettato per il cloud computing dinamico. Un indirizzo IP elastico è associato al tuo Account AWS. Si tratta di un indirizzo IP pubblico, raggiungibile da Internet. Se l'istanza in uso non dispone di un indirizzo IP pubblico, puoi associare un indirizzo IP elastico all'istanza per abilitare la comunicazione con Internet.

Gli esempi seguenti mostrano come:

- Descrivi una o più istanze utilizzando DescribeInstances.
- Acquisisci un indirizzo IP elastico utilizzando AllocateAddress.
- Associa un indirizzo IP elastico a un'istanza utilizzando AssociateAddress.
- Rilascia un indirizzo IP elastico utilizzando ReleaseAddress.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Descrivi un'istanza

Importazioni

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

use Aws\Ec2\Ec2Client;
```

Codice di esempio

```
$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";
    }
}
```

Assegna e associa un indirizzo

Importazioni

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

Codice di esempio

```
$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);
```

Rilascia un indirizzo

Importazioni

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

Codice di esempio

```
$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);
```

Utilizzo di regioni e zone di disponibilità per Amazon EC2 con AWS SDK per PHP la versione 3

Amazon EC2 è ospitato in diverse località in tutto il mondo. Queste località sono composte da AWS regioni e zone di disponibilità. Ogni regione è un'area geografica separata, con più località isolate note come zone di disponibilità. Amazon EC2 offre la possibilità di collocare istanze e dati in più posizioni.

Gli esempi seguenti mostrano come:

- Descrivi le zone di disponibilità disponibili per l'utilizzo DescribeAvailabilityZones.
- Descrivi le AWS regioni attualmente disponibili per l'utilizzo DescribeRegions.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Descrivere le zone di disponibilità

Importazioni

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

Codice di esempio

```
$ec2Client = new Aws\Ec2\Ec2Client([
```

```
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAavailabilityZones();

var_dump($result);
```

Descrivere le regioni

Importazioni

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

Codice di esempio

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

Utilizzo delle coppie di EC2 chiavi Amazon con AWS SDK per PHP la versione 3

Amazon EC2 utilizza la crittografia a chiave pubblica per crittografare e decrittografare le informazioni di accesso. La crittografia a chiave pubblica utilizza una chiave pubblica per crittografare i dati. A sua volta, il destinatario utilizza la chiave privata per decrittografare i dati. La chiave pubblica e quella privata sono note come coppia di chiavi.

Gli esempi seguenti mostrano come:

- Crea una coppia di chiavi RSA a 2048 bit utilizzando. CreateKeyPair
- Eliminare una key pair specificata utilizzando DeleteKeyPair.

- Descrivi una o più delle tue coppie di chiavi utilizzando DescribeKeyPairs.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creazione di una coppia di chiavi

Importazioni

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

Codice di esempio

```
$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);
```

Eliminare una coppia di chiavi

Importazioni

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

Codice di esempio

```
$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);
```

Descrivere coppie di chiavi

Importazioni

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

Codice di esempio

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

Lavorare con i gruppi di sicurezza in Amazon EC2 con AWS SDK per PHP la versione 3

Un gruppo EC2 di sicurezza Amazon funge da firewall virtuale che controlla il traffico per una o più istanze. A ciascun gruppo di sicurezza possono essere aggiunte regole che permettono il traffico da o verso le istanze associate. Puoi modificare le regole di un gruppo di sicurezza in qualsiasi momento. Le nuove regole vengono applicate automaticamente a tutte le istanze associate al gruppo di sicurezza.

Gli esempi seguenti mostrano come:

- Descrivi uno o più dei tuoi gruppi di sicurezza utilizzando DescribeSecurityGroups.
- Aggiungi una regola di ingresso a un gruppo di sicurezza utilizzando AuthorizeSecurityGroupIngress.
- Crea un gruppo di sicurezza utilizzando CreateSecurityGroup.
- Eliminare un gruppo di sicurezza utilizzando DeleteSecurityGroup.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Descrivere i gruppi di sicurezza

Importazioni

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

Codice di esempio

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);
```

```
$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

Aggiungi una regola di ingresso

Importazioni

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

Codice di esempio

```
$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);
```

Creazione di un gruppo di sicurezza

Importazioni

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

Codice di esempio

```
$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';
```

Eliminare un gruppo di sicurezza

Importazioni

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

Codice di esempio

```
$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);
```

Firma di una richiesta OpenSearch di ricerca Amazon Service con AWS SDK per PHP la versione 3

Amazon OpenSearch Service è un servizio gestito che semplifica l'implementazione, il funzionamento e la scalabilità di Amazon OpenSearch Service, un popolare motore di ricerca e analisi open source. OpenSearch Il servizio offre accesso diretto all'API OpenSearch di Amazon Service. Ciò significa che gli sviluppatori possono utilizzare gli strumenti con cui hanno familiarità, oltre a solide opzioni di sicurezza. Molti clienti di Amazon OpenSearch Service supportano la firma delle richieste, ma se utilizzi un client che non lo fa, puoi firmare richieste PSR-7 arbitrarie con i fornitori di credenziali e i firmatari integrati di. AWS SDK per PHP

Gli esempi seguenti mostrano come:

- Firma una richiesta con il protocollo di firma utilizzando SignatureV4 AWS .

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Firma di una richiesta OpenSearch di assistenza

OpenSearch Il servizio utilizza la versione 4 di Signature. Ciò significa che è necessario firmare le richieste utilizzando il nome di firma del servizio (`es`in questo caso) e la AWS regione del dominio del OpenSearch servizio. Un elenco completo delle regioni supportate dal OpenSearch servizio è disponibile nella pagina AWS Regioni ed endpoint del Riferimenti generali di Amazon Web Services. Tuttavia, in questo esempio, firmiamo le richieste relative a un dominio OpenSearch di servizio nella `us-west-2` regione.

Devi fornire le credenziali, cosa che puoi fare con la catena di provider predefinita dell'SDK o con qualsiasi forma di credenziale descritta in Credenziali per la versione 3. AWS SDK per PHP Dovrai anche disporre di una richiesta PSR-7 (nel codice di seguito si è presupposto il nome `$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 esempi utilizzando la AWS SDK per PHP versione 3

AWS Identity and Access Management (IAM) è un servizio Web che consente ai clienti di Amazon Web Services di gestire gli utenti e le autorizzazioni degli utenti in AWS. Il servizio è destinato alle organizzazioni con più utenti o sistemi nel cloud che utilizzano AWS prodotti. Con IAM, puoi gestire centralmente gli utenti, le credenziali di sicurezza come le chiavi di accesso e le autorizzazioni che controllano le AWS risorse a cui gli utenti possono accedere.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Argomenti

- Gestione delle chiavi di accesso IAM con AWS SDK per PHP la versione 3
- Gestione degli utenti IAM con AWS SDK per PHP la versione 3
- Utilizzo degli alias degli account IAM con la AWS SDK per PHP versione 3
- Utilizzo delle policy IAM con AWS SDK per PHP la versione 3
- Utilizzo dei certificati del server IAM con AWS SDK per PHP la versione 3

Gestione delle chiavi di accesso IAM con AWS SDK per PHP la versione 3

Gli utenti necessitano delle proprie chiavi di accesso per effettuare chiamate programmatiche a AWS. Per soddisfare questa esigenza, puoi creare, modificare, visualizzare o ruotare le chiavi di accesso (chiave di accesso IDs e chiavi di accesso segrete) per gli utenti IAM. Per impostazione predefinita, quando viene creata una chiave di accesso, il suo stato è Attivo. Pertanto, l'utente può utilizzare la chiave di accesso per le chiamate alle API.

Gli esempi seguenti mostrano come:

- Crea una chiave di accesso segreta e l'ID della chiave di accesso corrispondente utilizzando CreateAccessKey.
- Restituisce informazioni sulla chiave di accesso IDs associata a un utente IAM che utilizza ListAccessKeys.
- Recupera informazioni su quando una chiave di accesso è stata utilizzata l'ultima volta utilizzando GetAccessKeyLastUsed.
- Modifica lo stato di una chiave di accesso da Attivo a Inattivo o viceversa, utilizzando. UpdateAccessKey
- Elimina una coppia di key di accesso associata a un utente IAM utilizzando DeleteAccessKey.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creare una chiave di accesso

Importazioni

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

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

Codice di esempio

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

Elencare le chiavi di accesso

Importazioni

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

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

Codice di esempio

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

Ottieni informazioni sull'ultimo utilizzo di una chiave di accesso

Importazioni

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

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

Codice di esempio

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

Aggiornamento di una chiave di accesso

Importazioni

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

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

Codice di esempio

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

Eliminare una chiave di accesso

Importazioni

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

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

Codice di esempio

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

Gestione degli utenti IAM con AWS SDK per PHP la versione 3

Un utente IAM è un'entità creata AWS per rappresentare la persona o il servizio con cui lo utilizza per interagire AWS. Un utente in AWS è composto da un nome e da credenziali.

Gli esempi seguenti mostrano come:

- Crea un nuovo utente IAM utilizzando CreateUser.
- Elenca gli utenti IAM che utilizzano ListUsers.
- Aggiorna un utente IAM utilizzando UpdateUser.
- Recupera informazioni su un utente IAM utilizzando GetUser.
- Elimina un utente IAM utilizzando DeleteUser.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Crea un utente IAM

Importazioni

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

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

Codice di esempio

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

Elenca gli utenti IAM

Importazioni

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

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

Codice di esempio

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

Aggiorna un utente IAM

Importazioni

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

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

Codice di esempio

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

Ottieni informazioni su un utente IAM

Importazioni

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

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

Codice di esempio

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

Elimina un utente IAM

Importazioni

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

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

Codice di esempio

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

Utilizzo degli alias degli account IAM con la AWS SDK per PHP versione 3

Se desideri che l'URL della tua pagina di accesso contenga il nome della tua azienda o un altro identificativo descrittivo anziché il tuo Account AWS ID, puoi creare un alias per il tuo ID. Account AWS Se crei un Account AWS alias, l'URL della pagina di accesso cambia per incorporare l'alias.

Gli esempi seguenti mostrano come:

- Crea un alias utilizzando. CreateAccountAlias
- Elenca l'alias associato all' Account AWS utilizzo. ListAccountAliases
- Eliminare un alias utilizzando. DeleteAccountAlias

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creare un alias

Importazioni

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

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

Codice di esempio

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

Elencare gli alias di un account

Importazioni

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

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

Codice di esempio

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

Eliminare un alias

Importazioni

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

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

Codice di esempio

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

Utilizzo delle policy IAM con AWS SDK per PHP la versione 3

È possibile concedere autorizzazioni a un utente mediante la creazione di una policy. Una policy è un documento che elenca le azioni che un utente può eseguire e le risorse che tali azioni possono influenzare. Per impostazione predefinita, le azioni o le risorse che non sono esplicitamente

consentite sono negate. Le policy possono essere create e collegate a utenti, gruppi di utenti, ruoli assunti da utenti e risorse.

Gli esempi seguenti mostrano come:

- Crea una politica gestita utilizzando CreatePolicy.
- Allega una policy a un ruolo utilizzando AttachRolePolicy.
- Allega una policy a un utente utilizzando AttachUserPolicy.
- Allega una policy a un gruppo utilizzando AttachGroupPolicy.
- Rimuovi una politica di ruolo utilizzando DetachRolePolicy.
- Rimuovi una politica utente utilizzando DetachUserPolicy.
- Rimuovi una politica di gruppo utilizzando DetachGroupPolicy.
- Eliminare una politica gestita utilizzando DeletePolicy.
- Eliminare una politica di ruolo utilizzando DeleteRolePolicy.
- Eliminare una politica utente utilizzando DeleteUserPolicy.
- Eliminare una politica di gruppo utilizzando DeleteGroupPolicy.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creazione di una policy

Importazioni

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

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

Codice di esempio

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

Collegamento di una policy a un ruolo

Importazioni

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

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

Codice di esempio

```
$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());
```

```
}
```

Collegamento di una policy a un utente

Importazioni

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

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

Codice di esempio

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

Allega una politica a un gruppo

Importazioni

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

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

Codice di esempio

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

Scollegare una politica utente

Importazioni

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

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

Codice di esempio

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

Scollegare una politica di gruppo

Importazioni

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

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

Codice di esempio

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

Eliminazione di una policy

Importazioni

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

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

Codice di esempio

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

Eliminazione di una policy del ruolo

Importazioni

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

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

Codice di esempio

```
$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());
```

```
}
```

Eliminare una politica utente

Importazioni

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

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

Codice di esempio

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

Eliminazione di una policy di gruppo

Importazioni

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

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

Codice di esempio

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

Utilizzo dei certificati del server IAM con AWS SDK per PHP la versione 3

Per abilitare le connessioni HTTPS al tuo sito Web o alla tua applicazione AWS, devi disporre di un certificato del server SSL/TLS. Per utilizzare un certificato ottenuto da un provider esterno con il tuo sito Web o la tua applicazione AWS, devi caricare il certificato su IAM o importarlo in. AWS Certificate Manager

Gli esempi seguenti mostrano come:

- Elenca i certificati archiviati in IAM utilizzando ListServerCertificates.
- Recupera informazioni su un certificato utilizzando GetServerCertificate.
- Aggiorna un certificato utilizzando UpdateServerCertificate.
- Eliminare un certificato utilizzando DeleteServerCertificate.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Elencare i certificati del server

Importazioni

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

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

Codice di esempio

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

Recuperate un certificato del server

Importazioni

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

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

Codice di esempio

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

Aggiornamento di un certificato del server

Importazioni

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

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

Codice di esempio

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

Eliminazione di un certificato del server

Importazioni

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

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

Codice di esempio

```
$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 esempi utilizzando la AWS SDK per PHP versione 3

AWS Key Management Service (AWS KMS) è un servizio gestito che semplifica la creazione e il controllo delle chiavi di crittografia utilizzate per crittografare i dati. Per ulteriori informazioni AWS KMS, consulta la documentazione di Amazon KMS. Che tu stia scrivendo applicazioni PHP sicure o inviando dati ad altri AWS servizi, ti AWS KMS aiuta a mantenere il controllo su chi può usare le tue chiavi e accedere ai tuoi dati crittografati.

Tutto il codice di esempio per la AWS SDK per PHP versione 3 è disponibile GitHub qui.

Argomenti

- Lavorare con le chiavi utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3
- Crittografia e decrittografia delle chiavi di AWS KMS dati utilizzando la versione 3 AWS SDK per PHP
- Utilizzo delle politiche AWS KMS chiave utilizzando la AWS SDK per PHP versione 3
- Lavorare con le sovvenzioni utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3
- Lavorare con gli alias utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3

Lavorare con le chiavi utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3

Le risorse principali in () sono. AWS Key Management Service AWS KMSAWS KMS keys Puoi utilizzare una chiave KMS per crittografare i tuoi dati.

Gli esempi seguenti mostrano come:

- Crea una chiave KMS del cliente utilizzando. CreateKey
- Genera una chiave dati utilizzando GenerateDataKey.
- Visualizza una chiave KMS utilizzando DescribeKey.
- Ottieni la chiave IDs e le chiavi ARNS delle chiavi KMS utilizzando. ListKeys
- Abilita le chiavi KMS utilizzando. EnableKey
- Disabilita le chiavi KMS utilizzando. DisableKey

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'utilizzo di AWS Key Management Service (AWS KMS), consulta la Guida per gli AWS KMS sviluppatori.

Crea una chiave KMS

Per creare una chiave KMS, usa l'CreateKeyoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Generazione di una chiave di dati

Per generare una chiave di crittografia dei dati, utilizzare l'GenerateDataKeyoperazione. Questa operazione restituisce il testo normale e copie crittografate della chiave di dati creata. Specificare AWS KMS key in che modo generare la chiave dati.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Visualizza una chiave KMS

Per ottenere informazioni dettagliate su una chiave KMS, incluso l'Amazon Resource Name (ARN) della chiave KMS e lo stato della chiave, utilizza l'operazione. DescribeKey

`DescribeKey` non ottiene alias. Per ottenere gli alias, usa l'operazione. ListAliases

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Ottieni l'ID della chiave e la chiave ARNs di una chiave KMS

Per ottenere l'ID e l'ARN della chiave KMS, utilizzare l'operazione. ListAliases

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Abilita una chiave KMS

Per abilitare una chiave KMS disabilitata, usa l'EnableKeyoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Disattiva una chiave KMS

Per disabilitare una chiave KMS, usa l'DisableKeyoperazione. La disabilitazione di una chiave KMS ne impedisce l'utilizzo.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Crittografia e decrittografia delle chiavi di AWS KMS dati utilizzando la versione 3 AWS SDK per PHP

Le chiavi di dati sono chiavi di crittografia che possono essere usate per crittografare i dati, incluso grandi quantità di dati e altre chiavi di crittografia.

È possibile utilizzare AWS Key Management Service an's (AWS KMS) AWS KMS keyper generare, crittografare e decrittografare le chiavi di dati.

Gli esempi seguenti mostrano come:

- Crittografare una chiave di dati utilizzando Encrypt.
- Decrittografare una chiave di dati utilizzando Decrypt.
- Crittografa nuovamente una chiave dati con una nuova chiave KMS utilizzando. ReEncrypt

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'utilizzo di AWS Key Management Service (AWS KMS), consulta la Guida per gli AWS KMS sviluppatori.

Crittografa

L'operazione Encrypt è progettata per crittografare le chiavi dei dati, ma non viene utilizzata di frequente. Le GenerateDataKeyWithoutPlaintextoperazioni GenerateDataKeyand restituiscono chiavi di dati crittografate. È possibile utilizzare il `Encypt` metodo quando si spostano dati crittografati in

una nuova AWS regione e si desidera crittografare la relativa chiave dati utilizzando una chiave KMS nella nuova regione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$KmsClient = new Aws\Kms\KmsClient([
    'profile' => 'default',
    'version' => '2014-11-01',
    'region' => 'us-east-2'
]);

$keyId = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab';
$message = pack('c*', 1, 2, 3, 4, 5, 6, 7, 8, 9, 0);

try {
    $result = $KmsClient->encrypt([
        'KeyId' => $keyId,
        'Plaintext' => $message,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Decrypt

Per decrittografare una chiave di dati, utilizza l'operazione Decrypt.

`ciphertextBlob`Quello che specifichi deve essere il valore del `CiphertextBlob` campo proveniente da una risposta GenerateDataKeyo Encrypt. GenerateDataKeyWithoutPlaintext

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Ricrittografa

Per decrittografare una chiave dati crittografata e quindi ricrittografare immediatamente la chiave dati con una chiave KMS diversa, utilizza l'operazione. ReEncrypt Le operazioni vengono eseguite interamente sul lato server interno AWS KMS, quindi non espongono mai il testo in chiaro all'esterno. AWS KMS

Il `ciphertextBlob` valore specificato deve essere il valore del `CiphertextBlob` campo di una risposta o GenerateDataKeyGenerateDataKeyWithoutPlaintextEncrypt.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Utilizzo delle politiche AWS KMS chiave utilizzando la AWS SDK per PHP versione 3

Quando ne crei una AWS KMS key, stabilisci chi può utilizzare e gestire quella chiave KMS. Queste autorizzazioni sono contenute in un documento chiamato policy delle chiavi. Puoi utilizzare la politica chiave per aggiungere, rimuovere o modificare le autorizzazioni in qualsiasi momento per una chiave KMS gestita dal cliente, ma non puoi modificare la politica chiave per una chiave KMS AWS gestita. Per ulteriori informazioni, consulta Autenticazione e controllo degli accessi per. AWS KMS

Gli esempi seguenti mostrano come:

- Elenca i nomi delle politiche chiave che utilizzano ListKeyPolicies.
- Ottieni una politica chiave utilizzando GetKeyPolicy.

- Imposta una politica chiave utilizzando PutKeyPolicy.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'utilizzo di AWS Key Management Service (AWS KMS), consulta la Guida per gli AWS KMS sviluppatori.

Elenca tutte le politiche chiave

Per ottenere i nomi delle politiche chiave per una chiave KMS, usa l'`ListKeyPolicies`operazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Recupera una politica chiave

Per ottenere la politica chiave per una chiave KMS, usa l'`GetKeyPolicy`operazione.

`GetKeyPolicy` richiede un nome per la policy. A meno che tu non abbia creato una politica chiave quando hai creato la chiave KMS, l'unico nome di policy valido è quello predefinito. Scopri di più sulla politica delle chiavi predefinite nella Guida per gli AWS Key Management Service sviluppatori.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Imposta una politica chiave

Per stabilire o modificare una politica chiave per una chiave KMS, usa l'`PutKeyPolicy`operazione.

`PutKeyPolicy` richiede un nome per la policy. A meno che tu non abbia creato una politica chiave quando hai creato la chiave KMS, l'unico nome di politica valido è quello predefinito. Scopri di più sulla politica delle chiavi predefinite nella Guida per gli AWS Key Management Service sviluppatori.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Lavorare con le sovvenzioni utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3

Una concessione è un altro meccanismo per fornire le autorizzazioni. È un'alternativa alla politica chiave. Puoi utilizzare le sovvenzioni per concedere un accesso a lungo termine che consenta ai AWS mandanti di utilizzare i servizi AWS Key Management Service (AWS KMS) gestiti dai clienti. AWS KMS keys Per ulteriori informazioni, consulta Grants nella AWS KMS Developer Guide.AWS Key Management Service

Gli esempi seguenti mostrano come:

- Crea una sovvenzione per una chiave KMS utilizzando. CreateGrant
- Visualizza una concessione per una chiave KMS utilizzando. ListGrants
- Ritirare una sovvenzione per l'utilizzo di una chiave KMS. RetireGrant
- Revoca una concessione per una chiave KMS utilizzando. RevokeGrant

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'utilizzo di AWS Key Management Service (AWS KMS), consulta la Guida per gli AWS KMS sviluppatori.

Creazione di una concessione

Per creare una sovvenzione per un AWS KMS key, usa l'CreateGrantoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Visualizza una concessione

Per ottenere informazioni dettagliate sulle sovvenzioni concesse a an AWS KMS key, usa l'ListGrantsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Ritirare una borsa di studio

Per ritirare una sovvenzione per un AWS KMS key, usa l'RetireGrantoperazione. Ritira una concessione al termine del suo utilizzo.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Revocare una sovvenzione

Per revocare una concessione a un AWS KMS key, usa l'operazione. RevokeGrant Puoi revocare una concessione per negare esplicitamente le operazioni che dipendono da essa.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Lavorare con gli alias utilizzando l' AWS KMS API e la AWS SDK per PHP versione 3

AWS Key Management Service (AWS KMS) fornisce un nome di visualizzazione opzionale per un alias AWS KMS keychiamato.

Gli esempi seguenti mostrano come:

- Crea un alias utilizzando. CreateAlias
- Visualizza un alias utilizzando. ListAliases
- Aggiorna un alias utilizzando. UpdateAlias
- Elimina un alias utilizzando. DeleteAlias

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'utilizzo di AWS Key Management Service (AWS KMS), consulta la Guida per gli AWS KMS sviluppatori.

Creare un alias

Per creare un alias per una chiave KMS, usa l'CreateAliasoperazione. L'alias deve essere univoco nell'account e nella regione. AWS Se crei un alias per una chiave KMS che ha già un alias, `CreateAlias` crea un altro alias per la stessa chiave KMS. Non sostituisce l'alias esistente.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Visualizza un alias

Per elencare tutti gli alias del chiamante Account AWS e Regione AWS, usa l'operazione. ListAliases

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Aggiornare un alias

Per associare un alias esistente a una chiave KMS diversa, usa l'operazione. UpdateAlias

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un alias

Per eliminare un alias, utilizzare l'DeleteAliasoperazione. L'eliminazione di un alias non ha alcun effetto sulla chiave KMS sottostante.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Esempi di Amazon Kinesis con la versione 3 AWS SDK per PHP

Amazon Kinesis è un AWS servizio che raccoglie, elabora e analizza i dati in tempo reale. Configura i tuoi flussi di dati con Amazon Kinesis Data Streams o usa Amazon Data Firehose per inviare dati ad Amazon S3, OpenSearch Service, Amazon Redshift o Splunk.

Per ulteriori informazioni su Kinesis, consulta la documentazione di Amazon Kinesis.

Tutto il codice di esempio per la AWS SDK per PHP versione 3 è disponibile qui. GitHub

Argomenti

- Creazione di flussi di dati utilizzando l'API Kinesis Data Streams e la versione 3 AWS SDK per PHP
- Gestisci i frammenti di dati utilizzando l'API Kinesis Data Streams e la versione 3 AWS SDK per PHP
- Creazione di flussi di distribuzione utilizzando l'API Firehose e AWS SDK per PHP la versione 3

Creazione di flussi di dati utilizzando l'API Kinesis Data Streams e la versione 3 AWS SDK per PHP

Amazon Kinesis Data Streams consente di inviare dati in tempo reale. Crea un produttore di dati con Kinesis Data Streams che fornisca dati alla destinazione configurata ogni volta che aggiungi dati.

Per ulteriori informazioni, consulta Creating and Managing Streams nella Amazon Kinesis Developer Guide.

Gli esempi seguenti mostrano come:

- Crea un flusso di dati utilizzando. CreateAlias
- Ottieni dettagli su un singolo flusso di dati utilizzando DescribeStream.
- Elenca i flussi di dati esistenti utilizzando ListStreams.
- Invia dati a un flusso di dati esistente utilizzando PutRecord.
- Eliminare un flusso di dati utilizzando DeleteStream.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon Kinesis Developer Guide, consulta la Amazon Kinesis Data Streams Developer Guide.

Creare un flusso di dati utilizzando un flusso di dati Kinesis

Stabilisci un flusso di dati Kinesis in cui inviare le informazioni che devono essere elaborate da Kinesis utilizzando il seguente esempio di codice. Scopri di più sulla creazione e l'aggiornamento di flussi di dati nella Amazon Kinesis Developer Guide.

Per creare un flusso di dati Kinesis, usa l'CreateStreamoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Recupera un flusso di dati

Ottieni informazioni dettagliate su un flusso di dati esistente utilizzando il seguente codice di esempio. Per impostazione predefinita, restituisce informazioni sui primi 10 shard collegati al flusso di dati Kinesis specificato. Ricordati `StreamStatus` di controllare la risposta prima di scrivere dati su un flusso di dati Kinesis.

Per recuperare i dettagli su un flusso di dati Kinesis specificato, usa l'DescribeStreamoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca i flussi di dati esistenti collegati a Kinesis

Elenca i primi 10 flussi di dati provenienti dalla tua Account AWS regione selezionata. AWS Utilizza il `HasMoreStreams restituito per stabilire se ci sono più flussi associati al tuo account.

Per elencare i tuoi flussi di dati Kinesis, usa l'operazione. ListStreams

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Invia dati a un flusso di dati esistente

Dopo aver creato un flusso di dati, utilizza il seguente esempio per l'invio dei dati. Prima di inviare i dati, utilizza `DescribeStream` per controllare se i dati `StreamStatus` sono attivi.

Per scrivere un singolo record di dati in un flusso di dati Kinesis, usa l'PutRecordoperazione. Per scrivere fino a 500 record in un flusso di dati Kinesis, usa l'PutRecordsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un flusso di dati

Questo esempio illustra come eliminare un flusso di dati. L'eliminazione di un flusso di dati, inoltre, elimina tutti i dati inviati al flusso di dati. I flussi di dati Active Kinesis passano allo stato ELIMINAZIONE fino al completamento dell'eliminazione del flusso. Mentre è nello stato DELETING, il flusso continua a elaborare i dati.

Per eliminare un flusso di dati Kinesis, usa l'DeleteStreamoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Gestisci i frammenti di dati utilizzando l'API Kinesis Data Streams e la versione 3 AWS SDK per PHP

Amazon Kinesis Data Streams consente di inviare dati in tempo reale a un endpoint. La velocità del flusso di dati dipende dal numero di shard nel tuo flusso.

È possibile scrivere 1.000 record al secondo in un singolo shard. Ogni shard ha inoltre un limite di caricamento di 1 MiB al secondo. L'utilizzo viene calcolato e applicato per shard, quindi utilizza questi esempi per gestire la capacità dei dati e il costo del flusso.

Gli esempi seguenti mostrano come:

- Elenca gli shard in uno stream utilizzando. ListShards

- Aggiungi o riduci il numero di frammenti in uno stream utilizzando. UpdateShardCount

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon Kinesis Data Streams, consulta la Amazon Kinesis Data Streams Developer Guide.

Elenca i frammenti di flussi di dati

Elenca i dettagli di un massimo di 100 shard in un flusso specifico.

Per elencare gli shard in un flusso di dati Kinesis, usa l'ListShardsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Aggiungi altri shard di flussi di dati

Se hai bisogno di più shard dei flussi di dati, è possibile aumentare il tuo attuale numero di shard. Consigliamo di raddoppiare il tuo conteggio di shard nella fase di incremento. In questo modo viene eseguita una copia di ciascun shard attualmente disponibile per aumentare le tue capacità. È possibile raddoppiare il numero di shard solo due volte in un periodo di 24 ore.

Ricorda che la fatturazione per l'utilizzo di Kinesis Data Streams viene calcolata per shard, quindi quando la domanda diminuisce, ti consigliamo di dimezzare il numero di shard. Quando elimini gli shard, puoi solo ridurre la quantità di shard a metà del tuo attuale conteggio di shard.

Per aggiornare il numero di shard di un flusso di dati Kinesis, usa l'UpdateShardCountoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Creazione di flussi di distribuzione utilizzando l'API Firehose e AWS SDK per PHP la versione 3

Amazon Data Firehose consente di inviare dati in tempo reale ad altri AWS servizi, tra cui Amazon Kinesis Data Streams, Amazon S3, Amazon OpenSearch Service (Service) e OpenSearch Amazon Redshift, oppure a Splunk. Creare un produttore dati con flussi di distribuzione per fornire i dati alla destinazione configurata ogni volta che aggiungi dati.

Gli esempi seguenti mostrano come:

- Crea un flusso di distribuzione utilizzando. CreateDeliveryStream
- Ottieni dettagli su un singolo flusso di consegna utilizzando DescribeDeliveryStream.
- Elenca i tuoi flussi di consegna utilizzando ListDeliveryStreams.
- Invia dati a un flusso di consegna utilizzando PutRecord.
- Elimina un flusso di consegna utilizzando DeleteDeliveryStream.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon Data Firehose, consulta la Amazon Kinesis Data Firehose Developer Guide.

Crea un flusso di distribuzione utilizzando un flusso di dati Kinesis

Per stabilire un flusso di distribuzione che inserisca i dati in un flusso di dati Kinesis esistente, usa l'CreateDeliveryStreamoperazione.

Ciò consente agli sviluppatori di migrare i servizi Kinesis esistenti su Firehose.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Crea un flusso di distribuzione utilizzando un bucket Amazon S3

Per stabilire un flusso di distribuzione che inserisca i dati in un bucket Amazon S3 esistente, utilizza l'operazione. CreateDeliveryStream

Fornire i parametri di destinazione, come descritto in Parametri di destinazione. Quindi assicurati di concedere a Firehose l'accesso al tuo bucket Amazon S3, come descritto in Concedere a Kinesis Data Firehose l'accesso a una destinazione Amazon S3.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Crea un flusso di distribuzione utilizzando Service OpenSearch

Per stabilire un flusso di distribuzione Firehose che inserisca i dati in un cluster OpenSearch di servizi, utilizzate l'CreateDeliveryStreamoperazione.

Fornire i parametri di destinazione, come descritto in Parametri di destinazione. Assicurati di concedere a Firehose l'accesso al tuo cluster di OpenSearch servizi, come descritto in Concedere a Kinesis Data Firehose l'accesso a una destinazione Amazon ES.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Recupera un flusso di distribuzione

Per ottenere i dettagli su un flusso di distribuzione Firehose esistente, utilizzare l'DescribeDeliveryStreamoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca i flussi di distribuzione esistenti collegati a Kinesis Data Streams

Per elencare tutti i flussi di distribuzione Firehose esistenti che inviano dati a Kinesis Data Streams, utilizzare l'operazione. ListDeliveryStreams

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca i flussi di consegna esistenti che inviano dati ad altri servizi AWS

Per elencare tutti i flussi di distribuzione Firehose esistenti che inviano dati ad Amazon S3, Service OpenSearch o Amazon Redshift o a Splunk, utilizza l'operazione. ListDeliveryStreams

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Inviare dati a un flusso di distribuzione Firehose esistente

Per inviare dati tramite un flusso di distribuzione Firehose alla destinazione specificata, utilizzare l'PutRecordoperazione dopo aver creato un flusso di consegna Firehose.

Prima di inviare dati a un flusso di distribuzione Firehose, controllate `DescribeDeliveryStream` se il flusso di distribuzione è attivo.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un flusso di distribuzione Firehose

Per eliminare un flusso di distribuzione Firehose, utilizzare l'DeleteDeliveryStreamsoperazione. Ciò elimina anche tutti i dati inviati al flusso di distribuzione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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 esempi utilizzando la AWS SDK per PHP versione 3

AWS Elemental MediaConvert è un servizio di transcodifica video basato su file con funzionalità di livello broadcast. Puoi usarlo per creare risorse per la trasmissione e la distribuzione video-on-demand (VOD) su Internet. Per ulteriori informazioni, consulta la Guida per l'utente AWS Elemental MediaConvert.

L'API PHP per AWS Elemental MediaConvert è esposta tramite la classe *`AWS.MediaConvert`*client. Per ulteriori informazioni, consulta il riferimento `Class: AWS.MediaConvert`all'API.

Crea e gestisci lavori di transcodifica in AWS Elemental MediaConvert

In questo esempio, si utilizza la AWS SDK per PHP versione 3 per chiamare AWS Elemental MediaConvert e creare un processo di transcodifica. Prima di iniziare, devi caricare il video di input nel bucket Amazon S3 che hai fornito per lo storage di input. Per un elenco dei codec e contenitori di input video supportati, consulta Codecs e contenitori di input supportati nella Guida per l'utente.AWS Elemental MediaConvert

Gli esempi seguenti mostrano come:

- Crea lavori di transcodifica in. AWS Elemental MediaConvertCreateJob.
- Annulla un processo di transcodifica dalla AWS Elemental MediaConvert coda. CancelJob
- Recupera il codice JSON per un processo di transcodifica completato. GetJob
- Recupera un array JSON per un massimo di 20 dei lavori creati più di recente. ListJobs

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per accedere al MediaConvert client, crea un ruolo IAM che dia AWS Elemental MediaConvert accesso ai tuoi file di input e ai bucket Amazon S3 in cui sono archiviati i file di output. Per i dettagli, consulta Configurare le autorizzazioni IAM nella Guida per l'AWS Elemental MediaConvert utente.

Crea un cliente

Configuralo AWS SDK per PHP creando un MediaConvert client, con la regione per il tuo codice. In questo esempio, la regione è impostata su us-west-2.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

Codice di esempio

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

Definizione di un semplice processo di transcodifica

Crea il file JSON che definisce i parametri del processo di transcodifica.

Questi parametri sono dettagliati. È possibile utilizzare la AWS Elemental MediaConvert console per generare i parametri del lavoro JSON scegliendo le impostazioni del processo nella console e quindi selezionando Mostra lavoro JSON nella parte inferiore della sezione Job. Questo esempio illustra il JSON per un processo semplice.

Codice di esempio

```
$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"
    ]
];
```

Crea un processo.

Dopo aver creato il JSON dei parametri del processo, chiama il metodo createJob invocando un `AWS.MediaConvert service objecte` trasferendo i parametri. L'ID del processo creato viene restituito nei dati della risposta.

Codice di esempio

```
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";
}
```

Recupera un lavoro

Con il JobID restituito quando viene chiamato createjob, è possibile ottenere descrizioni dettagliate dei processi recenti in formato JSON.

Codice di esempio

```
$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";
}
```

Annullamento di un processo

Con il JobID restituito quando viene chiamato createjob, è possibile annullare un processo mentre è ancora in coda. Non è possibile annullare i processi che hanno già avviato la transcodifica.

Codice di esempio

```
$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";
}
```

Elenco dei lavori di transcodifica recenti

Crea il JSON dei parametri, inclusi i valori per specificare se ordinare l'elenco in ordine CRESCENTE o DECRESCENTE, l'ARN della coda dei processi da controllare e lo stato dei processi da includere. Vengono restituiti fino a 20 processi. Per recuperare i successivi 20 processi più recenti, utilizza la stringa nextToken restituita con il risultato.

Codice di esempio

```
$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";
}
```

Esempi di Amazon S3 con la AWS SDK per PHP versione 3

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) è un servizio Web che fornisce storage nel cloud altamente scalabile. Amazon S3 offre uno storage di oggetti facile da usare, con una semplice interfaccia di servizi Web per archiviare e recuperare qualsiasi quantità di dati da qualsiasi punto del Web.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Argomenti

- Creazione e utilizzo di bucket Amazon S3 con la versione 3 AWS SDK per PHP
- Gestione delle autorizzazioni di accesso ai bucket Amazon S3 con la versione 3 AWS SDK per PHP
- Configurazione dei bucket Amazon S3 con la versione 3 AWS SDK per PHP
- Utilizzo dei caricamenti multiparte di Amazon S3 con la versione 3 AWS SDK per PHP
- URL prefirmato Amazon S3 con versione 3 AWS SDK per PHP
- Amazon S3 prefirmato POSTs con la versione 3 AWS SDK per PHP
- Utilizzo di un bucket Amazon S3 come host Web statico con la versione 3 AWS SDK per PHP
- Utilizzo delle policy relative ai bucket di Amazon S3 con la versione 3 AWS SDK per PHP
- Utilizzo del punto di accesso S3 (ARNs la AWS SDK per PHP versione 3)
- Usa punti di accesso multiregionali Amazon S3 con la versione 3 AWS SDK per PHP

Creazione e utilizzo di bucket Amazon S3 con la versione 3 AWS SDK per PHP

Gli esempi seguenti mostrano come:

- Restituisce un elenco di bucket di proprietà del mittente autenticato della richiesta utilizzando. ListBuckets
- Crea un nuovo bucket usando. CreateBucket
- Aggiungi un oggetto a un bucket usando. PutObject

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Importazioni

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

use Aws\S3\S3Client;
```

Elenco di bucket

Crea un file PHP con il codice seguente. Per prima cosa crea un servizio client AWS.S3 che specifichi la regione e la AWS versione. Quindi chiama il `listBuckets` metodo, che restituisce tutti i bucket Amazon S3 di proprietà del mittente della richiesta come matrice di strutture di bucket.

Codice di esempio

```
$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";
}
```

Creazione di un bucket

Crea un file PHP con il codice seguente. Per prima cosa crea un servizio client AWS.S3 che specifichi la regione e la versione. AWS Quindi chiama il metodo `createBucket` con un array come parametro. L'unico campo obbligatorio è la chiave "Bucket", con un valore stringa per il nome del bucket che desideri creare. Tuttavia, puoi specificare la AWS regione con il campo ". CreateBucketConfiguration In caso di esito positivo, questo metodo restituisce il "Percorso" del bucket.

Codice di esempio

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
```

```
            'Bucket' => $bucketName,
        ]);
        return 'The bucket\'s location is: ' .
            $result['Location'] . '. ' .
            'The bucket\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheBucket()
{
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2006-03-01'
    ]);

    echo createBucket($s3Client, 'my-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();
```

Metti un oggetto in un secchio

Per aggiungere file al nuovo bucket, crea un file PHP con il seguente codice.

Nella riga di comando, esegui questo file e passa il nome del bucket in cui desideri caricare il tuo file come stringa, seguito dal percorso completo del file da caricare.

Codice di esempio

```
$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";
}
```

Gestione delle autorizzazioni di accesso ai bucket Amazon S3 con la versione 3 AWS SDK per PHP

Le liste di controllo degli accessi (ACLs) sono una delle opzioni di policy di accesso basate sulle risorse che puoi utilizzare per gestire l'accesso ai tuoi bucket e oggetti. È possibile utilizzare ACLs per concedere autorizzazioni di lettura/scrittura di base ad altri account. AWS Per ulteriori informazioni, consulta Gestire l'accesso con. ACLs

Gli esempi seguenti mostrano come:

- Ottieni la politica di controllo degli accessi per un bucket utilizzando GetBucketAcl.
- Imposta le autorizzazioni su un bucket utilizzando ACLs, using. PutBucketAcl

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Ottieni e imposta una politica per la lista di controllo degli accessi

Importazioni

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

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

Codice di esempio

```

// 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";
}
```

Configurazione dei bucket Amazon S3 con la versione 3 AWS SDK per PHP

La funzionalità CORS (Cross-Origin Resource Sharing, condivisione delle risorse multiorigine) definisce un metodo con cui le applicazioni Web dei clienti caricate in un dominio possono interagire con le risorse situate in un dominio differente. Con il supporto CORS in Amazon S3, puoi creare ricche applicazioni Web lato client con Amazon S3 e consentire selettivamente l'accesso multiorigine alle tue risorse Amazon S3.

Per ulteriori informazioni sull'utilizzo della configurazione CORS con un bucket Amazon S3, consulta Cross-Origin Resource Sharing (CORS).

Gli esempi seguenti mostrano come:

- Ottieni la configurazione CORS per un bucket utilizzando. GetBucketCors
- Imposta la configurazione CORS per un bucket utilizzando. PutBucketCors

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Ottieni la configurazione CORS

Crea un file PHP con il codice seguente. Innanzitutto, crea un servizio client AWS.S3, quindi richiama il metodo `getBucketCors` e specifica il bucket con la configurazione CORS desiderata.

L'unico parametro obbligatorio è il nome del bucket selezionato. Se il bucket ha attualmente una configurazione CORS, tale configurazione viene restituita da Amazon S3 come oggetto. CORSRules

Importazioni

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

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

Codice di esempio

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

Imposta la configurazione CORS

Crea un file PHP con il codice seguente. Innanzitutto, crea un servizio client AWS.S3. Quindi chiamate il `putBucketCors` metodo e specificate il bucket di cui impostare la configurazione CORS, quindi CORSConfiguration come oggetto JSON. CORSRules

Importazioni

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

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

Codice di esempio

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

Utilizzo dei caricamenti multiparte di Amazon S3 con la versione 3 AWS SDK per PHP

Con una singola operazione `PutObject`, puoi caricare oggetti di una dimensione massima di 5 GB. Tuttavia, tramite i metodi di caricamento in più parti (ad esempio, `CreateMultipartUpload`, `UploadPart`, `CompleteMultipartUpload`, `AbortMultipartUpload`), puoi caricare oggetti di dimensioni comprese tra 5 MB e 5 TB.

Gli esempi seguenti mostrano come:

- Carica un oggetto su Amazon S3, utilizzando. ObjectUploader
- Crea un caricamento multiparte per un oggetto Amazon S3 utilizzando. MultipartUploader
- Copia oggetti da una posizione Amazon S3 a un'altra utilizzando. ObjectCopier

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Caricatore di oggetti

Se non sei sicuro che sia `PutObject` o `MultipartUploader` meno la soluzione migliore per l'attività, usa. `ObjectUploader` `ObjectUploader`carica un file di grandi dimensioni su Amazon S3 utilizzando `PutObject` `MultipartUploader` o, in base alla dimensione del payload.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\ObjectUploader;
use Aws\S3\S3Client;
```

Codice di esempio

```
// 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);
```

Configurazione

Il costruttore dell'oggetto `ObjectUploader` accetta i seguenti argomenti:

`$client`

L'oggetto `Aws\ClientInterface` da utilizzare per eseguire i trasferimenti. Deve essere un'istanza di `Aws\S3\S3Client`.

`$bucket`

(`string`, obbligatorio) Nome del bucket in cui è in corso il caricamento dell'oggetto.

`$key`

(`string`, obbligatorio) Chiave da utilizzare per l'oggetto in fase di caricamento.

`$body`

(`mixed`, obbligatorio) Dati dell'oggetto da caricare. Può essere una `StreamInterface` risorsa di flusso PHP o una stringa di dati da caricare.

`$acl`

(`string`) Lista di controllo accessi (ACL) da impostare sull'oggetto in fase di caricamento. Gli oggetti sono privati per impostazione predefinita.

`$options`

Un array associativo delle opzioni di configurazione del caricamento in più parti. Le seguenti opzioni di configurazione sono valide:

`add_content_md5`

(`bool`) Imposta su true per calcolare automaticamente il MD5 checksum per il caricamento.

`mup_threshold`

(`int`, impostazione predefinita:`int(16777216)`) Il numero di byte per la dimensione del file. Se la dimensione del file supera questo limite, viene utilizzato un caricamento in più parti.

`before_complete`

(`callable`) Callback da richiamare prima dell'operazione `CompleteMultipartUpload`. Il callback dovrebbe avere una firma di funzione simile a:. `function (Aws\Command $command) {...}` Consulta il riferimento CompleteMultipartUpload API per i parametri che puoi aggiungere all'`CommandInterface`oggetto.

`before_initiate`

(`callable`) Callback da richiamare prima dell'operazione `CreateMultipartUpload`. Il callback dovrebbe avere una firma di funzione simile a:`function (Aws\Command`

`$command) {...}`. L'SDK richiama questo callback se la dimensione del file supera il valore. `mup_threshold` Consulta il riferimento all'CreateMultipartUpload API per i parametri che puoi aggiungere all'oggetto. `CommandInterface`

`before_upload`

(`callable`) Richiamata da richiamare prima di `PutObject` qualsiasi `UploadPart` operazione. Il callback dovrebbe avere una firma di funzione simile a:. `function (Aws\Command $command) {...}` L'SDK richiama questo callback se la dimensione del file è inferiore o uguale al valore. `mup_threshold` Consulta il riferimento all'PutObject API per i parametri che puoi applicare alla richiesta. `PutObject` Per i parametri che si applicano a una `UploadPart` richiesta, consulta il riferimento all'UploadPart API. L'SDK ignora qualsiasi parametro non applicabile all'operazione rappresentata dall'`CommandInterface`oggetto.

`concurrency`

(`int`, impostazione predefinita: `int(3)`) Numero massimo di operazioni `UploadPart` simultanee consentite durante il caricamento in più parti.

`part_size`

(`int`, impostazione predefinita: `int(5242880)`) Dimensioni, in byte, della parte da utilizzare durante il caricamento in più parti. Il valore deve essere compreso tra 5 MB e 5 GB, inclusi.

`state`

(`Aws\Multipart\UploadState`) Un oggetto che rappresenta lo stato del caricamento in più parti e che viene utilizzato per riprendere un caricamento precedente. Quando viene fornita questa opzione, gli `$key` argomenti `$bucket` and e l'`part_size`opzione vengono ignorati.

`params`

Un array associativo che fornisce opzioni di configurazione per ogni sottocomando. Per esempio:

```
new ObjectUploader($bucket, $key, $body, $acl, ['params' => ['CacheControl'
 => <some_value>])
```

MultipartUploader

I caricamenti in più parti sono concepiti per migliorare l'esperienza di caricamento degli oggetti di maggiori dimensioni. Consentono di caricare gli oggetti in parti indipendenti, in qualsiasi ordine e in parallelo.

I clienti di Amazon S3 sono incoraggiati a utilizzare caricamenti multiparte per oggetti di dimensioni superiori a 100 MB.

MultipartUploader oggetto

L'SDK contiene un oggetto `MultipartUploader` speciale che semplifica il processo di caricamento in più parti.

Importazioni

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Codice di esempio

```
$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";
}
```

L'uploader crea un generatore di dati della parte, basato sulla sorgente e sulla configurazione specificate, e tenta di caricare tutte le parti. Se il caricamento di alcune parti non riesce, l'uploader

continua con il caricamento delle parti successive fino a quando non vengono letti tutti i dati di origine. Successivamente, l'uploader riprova a caricare le parti non riuscite o genera un'eccezione contenente informazioni sulle parti che non è stato possibile caricare.

Personalizzazione di un caricamento in più parti

Puoi impostare opzioni personalizzate nelle operazioni `CreateMultipartUpload`, `UploadPart` e `CompleteMultipartUpload` eseguite dall'uploader in più parti tramite i callback trasmessi al relativo costruttore.

Importazioni

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

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

Codice di esempio

```
// 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';
    },
]);
```

Raccolta manuale dei rifiuti tra i caricamenti delle parti

Se caricamenti di grandi dimensioni causano il raggiungimento del limite di memoria, la causa è da ricercarsi nei riferimenti ciclici generati dall'SDK che non sono ancora stati raccolti dal garbage collector PHP quando il limite di memoria è stato raggiunto. L'invocazione manuale dell'algoritmo di raccolta tra operazioni può consentire la raccolta dei cicli prima del raggiungimento di tale limite. L'esempio seguente richiama l'algoritmo di raccolta utilizzando un callback prima del caricamento di ogni parte. L'invocazione di garbage collector comporta costi di prestazioni e l'uso ottimale dipende dal caso d'uso e dall'ambiente.

```
$uploader = new MultipartUploader($client, $source, [
   'bucket' => 'your-bucket',
   'key' => 'your-key',
   'before_upload' => function(\Aws\Command $command) {
      gc_collect_cycles();
   }
]);
```

Recupero da errori

Quando si verifica un errore durante il processo di caricamento in più parti, viene generata una `MultipartUploadException`. Questa eccezione consente l'accesso all'oggetto `UploadState`, contenente le informazioni sull'avanzamento del caricamento in più parti. L'oggetto `UploadState` può essere utilizzato per riprendere un caricamento che non è stato possibile completare.

Importazioni

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Codice di esempio

```
// 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);
}
```

La ripresa di un caricamento da un oggetto `UploadState` tenta di caricare le parti che non sono già state caricate. L'oggetto stato tiene traccia delle parti mancanti, anche se non sono consecutive. L'uploader legge o cerca nel file sorgente fornito negli intervalli di byte appartenenti alle parti che devono ancora essere caricate.

Gli oggetti `UploadState` sono serializzabili, quindi puoi riprendere un caricamento anche in un processo diverso. Puoi ottenere inoltre l'oggetto `UploadState`, anche quando non stai gestendo un'eccezione, richiamando `$uploader->getState()`.

> **Important**
>
> I flussi trasmessi come sorgente a `MultipartUploader` non vengono riavvolti automaticamente prima del caricamento. Se utilizzi un flusso al posto di un percorso di file in un loop simile a quello dell'esempio precedente, ripristina la variabile `$source` all'interno del blocco `catch`.

Importazioni

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Codice di esempio

```
// 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);
```

Interruzione di un caricamento in più parti

Un caricamento in più parti può essere interrotto recuperando l'`UploadId` contenuto nell'oggetto `UploadState` e passandolo a `abortMultipartUpload`.

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

Caricamenti asincroni multiparte

La chiamata di `upload()` sul `MultipartUploader` corrisponde a una richiesta di blocco. Se stai lavorando in un contesto asincrono, potresti ricevere una promessa per il caricamento in più parti.

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

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

Codice di esempio

```
// 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();
```

Configurazione

Il costruttore dell'oggetto `MultipartUploader` accetta i seguenti argomenti:

`$client`

L'oggetto `Aws\ClientInterface` da utilizzare per eseguire i trasferimenti. Deve essere un'istanza di `Aws\S3\S3Client`.

`$source`

I dati di origine in fase di caricamento. Può essere un percorso o un URL (ad esempio, `/path/to/file.jpg`), un gestore di risorse (ad esempio, `fopen('/path/to/file.jpg', 'r')`) o un'istanza di un flusso PSR-7.

`$config`

Un array associativo delle opzioni di configurazione del caricamento in più parti.

Le seguenti opzioni di configurazione sono valide:

`acl`

(`string`) Lista di controllo accessi (ACL) da impostare sull'oggetto in fase di caricamento. Gli oggetti sono privati per impostazione predefinita.

`before_complete`

(`callable`) Callback da richiamare prima dell'operazione `CompleteMultipartUpload`. Il callback deve disporre di una firma della funzione, come `function (Aws\Command $command) {...}`.

`before_initiate`

(`callable`) Callback da richiamare prima dell'operazione `CreateMultipartUpload`. Il callback deve disporre di una firma della funzione, come `function (Aws\Command $command) {...}`.

`before_upload`

(`callable`) Callback da richiamare prima delle operazioni `UploadPart`. Il callback deve disporre di una firma della funzione, come `function (Aws\Command $command) {...}`.

`bucket`

(`string`, obbligatorio) Nome del bucket in cui è in corso il caricamento dell'oggetto.

`concurrency`

(`int`, impostazione predefinita: `int(5)`) Numero massimo di operazioni `UploadPart` simultanee consentite durante il caricamento in più parti.

`key`

(`string`, obbligatorio) Chiave da utilizzare per l'oggetto in fase di caricamento.

`part_size`

(`int`, impostazione predefinita: `int(5242880)`) Dimensioni, in byte, della parte da utilizzare durante il caricamento in più parti. Le dimensioni devono essere comprese tra 5 MB e 5 GB, inclusi.

`state`

(`Aws\Multipart\UploadState`) Un oggetto che rappresenta lo stato del caricamento in più parti e che viene utilizzato per riprendere un caricamento precedente. Quando questa opzione è disponibile, le opzioni `bucket`, `key` e `part_size` vengono ignorate.

`add_content_md5`

(`boolean`) Imposta su true per calcolare automaticamente il checksum per il MD5 caricamento.

`params`

Un array associativo che fornisce opzioni di configurazione per ogni sottocomando. Per esempio:

```
new MultipartUploader($client, $source, ['params' => ['CacheControl'
 => <some_value>]])
```

Copie in più parti

Include AWS SDK per PHP anche un `MultipartCopy` oggetto che viene utilizzato in modo simile a`MultipartUploader`, ma è progettato per copiare oggetti di dimensioni comprese tra 5 GB e 5 TB all'interno di Amazon S3.

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

```
use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
use Aws\S3\S3Client;
```

Codice di esempio

```
// 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 prefirmato Amazon S3 con versione 3 AWS SDK per PHP

È possibile autenticare alcuni tipi di richieste passando le informazioni necessarie come parametri di una stringa di query invece di utilizzare l'intestazione Autorizzazione HTTP. Ciò è utile per consentire l'accesso diretto tramite browser di terze parti ai dati privati di Amazon S3, senza inoltrare la richiesta tramite proxy. L'idea è quella di creare una richiesta «prefirmata» e codificarla come URL utilizzabile da un altro utente. È inoltre possibile limitare una richiesta prefirmata specificando un periodo di scadenza.

Crea un URL prefirmato per una richiesta HTTP GET

Il seguente esempio di codice mostra come creare un URL prefirmato per una richiesta HTTP GET utilizzando l'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;
```

Il riferimento all'API per il `createPresignedRequest` metodo fornisce ulteriori dettagli.

Qualcun altro può utilizzare il `$presignedUrl` valore per recuperare l'oggetto entro l'ora successiva. Quando viene effettuata la richiesta HTTP GET, ad esempio utilizzando un browser, al servizio S3 sembra che la chiamata provenga dall'utente che ha creato l'URL prefirmato.

Crea un URL prefirmato per una richiesta HTTP PUT

Il seguente esempio di codice mostra come creare un URL prefirmato per una richiesta HTTP PUT utilizzando l'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();
```

Qualcun altro può ora utilizzare l'URL prefirmato in una richiesta HTTP PUT per caricare un 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 prefirmato POSTs con la versione 3 AWS SDK per PHP

Proprio come quelli prefirmati URLs, i prefirmati POSTs consentono di concedere l'accesso in scrittura a un utente senza fornire loro AWS credenziali. I moduli POST prefirmati possono essere creati con l'aiuto di un'istanza di AWSS3 V4. PostObject

Gli esempi seguenti mostrano come:

- Ottieni i dati per un modulo di caricamento S3 Object POST utilizzando V4. PostObject

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

> **Note**
>
> `PostObjectV4`non funziona con credenziali provenienti da. AWS IAM Identity Center

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto in. Credenziali Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Crea PostObject V4

Per creare un'istanza di `PostObjectV4` è necessario fornire quanto segue:

- istanza di `Aws\S3\S3Client`
- bucket
- array associativo dei campi di input del modulo
- serie di condizioni delle politiche (vedi Policy Construction nella Guida per l'utente di Amazon Simple Storage Service)

- stringa relativa al periodo di scadenza per la policy (opzionale, il valore predefinito è un'ora).

Importazioni

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

Codice di esempio

```
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>
```

Utilizzo di un bucket Amazon S3 come host Web statico con la versione 3 AWS SDK per PHP

È possibile ospitare un sito Web statico su Amazon S3. Per ulteriori informazioni, consulta Hosting di un sito Web statico su Amazon S3.

Gli esempi seguenti mostrano come:

- Ottieni la configurazione del sito Web per un bucket utilizzando. GetBucketWebsite
- Imposta la configurazione del sito Web per un bucket utilizzando. PutBucketWebsite
- Rimuovi la configurazione del sito Web da un bucket utilizzando. DeleteBucketWebsite

Tutto il codice di esempio per la AWS SDK per PHP versione 3 è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configura le tue AWS credenziali. Vedi Credenziali per la AWS SDK per PHP versione 3.

Ottieni, imposta ed elimina la configurazione del sito Web per un bucket

Importazioni

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

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

Codice di esempio

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

// Retrieving the Bucket Website Configuration
$bucket = 'my-s3-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}

// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
        ],
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ]
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
```

```
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Utilizzo delle policy relative ai bucket di Amazon S3 con la versione 3 AWS SDK per PHP

Puoi utilizzare una bucket policy per concedere l'autorizzazione alle tue risorse Amazon S3. Per ulteriori informazioni, consulta Utilizzo delle policy per i bucket e delle policy utente.

Gli esempi seguenti mostrano come:

- Restituisci la politica per un bucket specificato utilizzando. GetBucketPolicy
- Sostituisci una politica su un bucket utilizzando. PutBucketPolicy
- Elimina una policy da un bucket utilizzando. DeleteBucketPolicy

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Ottieni, elimina e sostituisci una politica su un bucket

Importazioni

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

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

Codice di esempio

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

$bucket = 'my-s3-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
    echo "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deletes the policy from the bucket
try {
    $resp = $s3Client->deleteBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Replaces a policy on the bucket
try {
    $resp = $s3Client->putBucketPolicy([
        'Bucket' => $bucket,
        'Policy' => 'foo policy',
    ]);
    echo "Succeed in put a policy on bucket: " . $bucket . "\n";
} catch (AwsException $e) {
```

```
    // Display error message
    echo $e->getMessage();
    echo "\n";
}
```

Utilizzo del punto di accesso S3 (ARNs la AWS SDK per PHP versione 3)

S3 ha introdotto i punti di accesso, un nuovo modo di interagire con i bucket S3. I punti di accesso possono avere criteri e configurazioni univoci applicati ad essi piuttosto che direttamente al bucket. AWS SDK per PHP Consente di utilizzare il punto di accesso ARNs nel campo del bucket per le operazioni API invece di specificare il nome del bucket in modo esplicito. Maggiori dettagli su come funzionano e ARNs come funzionano i punti di accesso S3 sono disponibili qui. Gli esempi seguenti mostrano come:

- GetObjectUtilizzalo con un punto di accesso ARN per recuperare un oggetto da un bucket.
- PutObjectUtilizzalo con un punto di accesso ARN per aggiungere un oggetto a un bucket.
- Configurare il client S3 per utilizzare la regione ARN anziché la regione client.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Importazioni

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

use Aws\S3\S3Client;
```

Ottieni oggetto

Per prima cosa crea un servizio client AWS.S3 che specifichi la regione e la AWS versione. Quindi chiama il `getObject` metodo con la tua chiave e un ARN del punto di accesso S3 nel campo `Bucket`, che recupererà l'oggetto dal bucket associato a quel punto di accesso.

Codice di esempio

```
$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'
]);
```

Metti un oggetto in un secchio

Per prima cosa crea un servizio client AWS.S3 che specifichi la regione e la versione. AWS Quindi chiama il metodo `putObject` con la chiave desiderata, il corpo o il file sorgente e un punto di accesso S3 nel campo `Bucket`, che metterà l'oggetto nel bucket associato a quel punto di accesso.

Codice di esempio

```
$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'
]);
```

Configurare il client S3 per utilizzare la regione ARN anziché la regione client

Quando si utilizza un ARN del punto di accesso S3 in un'operazione client S3, per impostazione predefinita il client si assicurerà che la regione ARN corrisponda alla regione client, generando un'eccezione in caso contrario. Questo comportamento può essere modificato per accettare la regione ARN sulla regione client impostando l'opzione di configurazione `use_arn_region` su `true`. Per impostazione predefinita, l'opzione è impostata su `false`.

Codice di esempio

```
$s3 = new S3Client([
```

```
    'version'        => 'latest',
    'region'         => 'us-west-2',
    'use_arn_region' => true
]);
```

Il client controllerà anche una variabile d'ambiente e un'opzione del file di configurazione, nel seguente ordine di priorità:

1. L'opzione client `use_arn_region`, come nell'esempio precedente.
2. Nella variabile di ambiente `AWS_S3_USE_ARN_REGION`.

```
export AWS_S3_USE_ARN_REGION=true
```

1. La variabile di configurazione `s3_use_arn_region` nel file di configurazione AWS condiviso (per impostazione predefinita in). `~/.aws/config`

```
[default]
s3_use_arn_region = true
```

Usa punti di accesso multiregionali Amazon S3 con la versione 3 AWS SDK per PHP

Gli access point multiregionali di Amazon Simple Storage Service (S3) forniscono un endpoint globale per il routing del traffico di richieste Amazon S3 tra di loro. Regioni AWS

Puoi creare punti di accesso multiregionali utilizzando l'SDK for PHP, AWS un altro SDK, la console S3 o la CLI, AWS

> **Important**
>
> Per utilizzare punti di accesso multiregionali con l'SDK for PHP, nell'ambiente PHP deve essere installata l'estensione Common Runtime AWS (AWS CRT).

Quando crei un punto di accesso multiregionale, Amazon S3 genera un Amazon Resource Name (ARN) con il seguente formato:

`arn:aws:s3::`*`account-id`*`:accesspoint/`*`MultiRegionAccessPoint_alias`*

È possibile utilizzare l'ARN generato al posto del nome del bucket per `getObject()` i metodi e. `putObject()`

```
<?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";
```

Gestione dei segreti utilizzando l'API Secrets Manager e la AWS SDK per PHP versione 3

AWS Secrets Manager archivia e gestisce segreti condivisi come password, chiavi API e credenziali del database. Con il servizio Secrets Manager, gli sviluppatori possono sostituire le credenziali codificate nel codice distribuito con una chiamata incorporata a Secrets Manager.

Secrets Manager supporta nativamente la rotazione automatica pianificata delle credenziali per i database Amazon Relational Database Service (Amazon RDS), aumentando la sicurezza delle applicazioni. Secrets Manager può anche ruotare senza problemi i segreti per altri database e servizi di terze parti utilizzandoli per AWS Lambda implementare dettagli specifici del servizio.

Gli esempi seguenti mostrano come:

- Crea un segreto usando. CreateSecret
- Recupera un segreto usando GetSecretValue.
- Elenca tutti i segreti archiviati da Secrets Manager utilizzando ListSecrets.
- Ottieni dettagli su un segreto specificato utilizzando DescribeSecret.
- Aggiorna un segreto specificato utilizzando PutSecretValue.
- Imposta una rotazione segreta usando RotateSecret.
- Contrassegna un segreto per l'eliminazione utilizzando DeleteSecret.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Crea un segreto in Secrets Manager

Per creare un segreto in Secrets Manager, usa l'CreateSecretoperazione.

In questo esempio, un nome utente e una password vengono memorizzati come una stringa JSON.

Importazioni

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

```
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Recupera un segreto da Secrets Manager

Per recuperare il valore di un segreto archiviato in Secrets Manager, utilizzare l'GetSecretValueoperazione.

Nell'esempio seguente, `secret` è una stringa che contiene il valore memorizzato. Se il valore per `username` è <<USERNAME>> e il valore per `password` è<<PASSWORD>>, l'output di `secret` è:

```
{"username":"<<USERNAME>>","password":"<<PASSWORD>>"}
```

Utilizzare `json_decode($secret, true)` per accedere ai valori dell'array.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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;
```

Elenca i segreti archiviati in Secrets Manager

Ottieni un elenco di tutti i segreti archiviati da Secrets Manager utilizzando l'ListSecretsoperazione.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Recupera i dettagli di un segreto

I segreti archiviati contengono metadati sulle regole di rotazione, la data dell'ultimo accesso o modifica, i tag creati dall'utente e l'Amazon Resource Name (ARN). Per ottenere i dettagli di un segreto specificato archiviato in Secrets Manager, utilizzare l'DescribeSecretoperazione.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Come aggiornare il valore del segreto

Per memorizzare un nuovo valore segreto crittografato in Secrets Manager, utilizzare l'PutSecretValueoperazione.

Viene creata una nuova versione del segreto. Se una versione del segreto esiste già, aggiungi il parametro `VersionStages` con il valore in `AWSCURRENT` per garantire che il nuovo valore viene utilizzato quando viene recuperato.

Importazioni

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
```

```
}
```

Ruota il valore in un segreto esistente in Secrets Manager

Per ruotare il valore di un segreto esistente archiviato in Secrets Manager, utilizzate una funzione di rotazione Lambda e RotateSecretl'operazione.

Prima di iniziare, crea una funzione Lambda per ruotare il tuo segreto. Il AWS Code Sample Catalog contiene attualmente diversi esempi di codice Lambda per la rotazione delle credenziali del database Amazon RDS.

> **Note**
>
> Per ulteriori informazioni sulla rotazione dei segreti, consulta Rotating Your Secrets nella Guida per l'utente AWS Secrets Manager. AWS Secrets Manager

Dopo aver impostato la funzione Lambda, configura una nuova rotazione segreta.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Quando è configurata una rotazione, è possibile implementare una rotazione utilizzando l'RotateSecretoperazione.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un segreto da Secrets Manager

Per rimuovere un segreto specificato da Secrets Manager, utilizzare l'DeleteSecretoperazione. Per evitare l'eliminazione accidentale di un segreto, al segreto viene aggiunto automaticamente un DeletionDate timbro che specifica una finestra di tempo di ripristino in cui è possibile annullare l'eliminazione. Se non si specifica l'ora per la finestra di ripristino, il periodo di tempo predefinito è di 30 giorni.

Importazioni

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Informazioni correlate

Gli AWS SDK per PHP esempi utilizzano le seguenti operazioni REST dell'API Reference: AWS Secrets Manager

- CreateSecret
- GetSecretValue
- ListSecrets
- DescribeSecret
- PutSecretValue
- RotateSecret
- DeleteSecret

Per ulteriori informazioni sull'utilizzo AWS Secrets Manager, consulta la Guida AWS Secrets Manager per l'utente.

Esempi di Amazon SES con la AWS SDK per PHP versione 3

Amazon Simple Email Service (Amazon SES) Simple Email Service (Amazon SES) è una piattaforma di posta elettronica che offre un modo semplice e conveniente per inviare e ricevere e-mail utilizzando i propri indirizzi e-mail e domini. Per ulteriori informazioni su Amazon SES, consulta la Amazon SES Developer Guide.

AWS offre due versioni del servizio Amazon SES e, di conseguenza, l'SDK for PHP offre due versioni del client: e SESv2Client. SesClient Le funzionalità dei client si sovrappongono in molti casi, sebbene il modo in cui i metodi vengono chiamati o i risultati possano differire. I due offrono APIs anche funzionalità esclusive, quindi puoi utilizzare entrambi i client per accedere a tutte le funzionalità.

Gli esempi in questa sezione utilizzano tutti l'originale,`SesClient`.

Tutto il codice di esempio per la AWS SDK per PHP versione 3 è disponibile qui GitHub.

Argomenti

- Verifica delle identità e-mail utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3
- Creazione di modelli di e-mail personalizzati utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3
- Gestione dei filtri e-mail tramite l'API Amazon SES e la AWS SDK per PHP versione 3
- Creazione e gestione di regole e-mail utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3
- Monitoraggio dell'attività di invio tramite l'API Amazon SES e la AWS SDK per PHP versione 3

- Autorizzazione dei mittenti tramite l'API Amazon SES e la versione 3 AWS SDK per PHP

Verifica delle identità e-mail utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3

Quando inizi a utilizzare per la prima volta il tuo account Amazon Simple Email Service (Amazon SES), tutti i mittenti e i destinatari devono essere verificati nella AWS stessa regione a cui stai inviando le e-mail. Per ulteriori informazioni sull'invio di e-mail, consulta Invio di e-mail con Amazon SES.

Gli esempi seguenti mostrano come:

- Verifica un indirizzo e-mail utilizzando. VerifyEmailIdentity
- Verifica un dominio e-mail utilizzando VerifyDomainIdentity.
- Elenca tutti gli indirizzi e-mail utilizzando ListIdentities.
- Elenca tutti i domini di posta elettronica che utilizzano ListIdentities.
- Rimuovi un indirizzo email utilizzando DeleteIdentity.
- Rimuovi un dominio e-mail utilizzando DeleteIdentity.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Verifica un indirizzo e-mail

Amazon SES può inviare e-mail solo da indirizzi e-mail o domini verificati. Verificando un indirizzo e-mail, dimostri di essere il proprietario di quell'indirizzo e di voler consentire ad Amazon SES di inviare e-mail da quell'indirizzo.

Quando esegui il seguente esempio di codice, Amazon SES invia un'e-mail all'indirizzo specificato. Quando tu (o il destinatario dell'e-mail) fai clic sul link contenuto nel messaggio e-mail, l'indirizzo è verificato.

Per aggiungere un indirizzo e-mail al tuo account Amazon SES, utilizza l'VerifyEmailIdentityoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Verifica un dominio e-mail

Amazon SES può inviare e-mail solo da indirizzi e-mail o domini verificati. Verificando un dominio, dimostri di essere il proprietario di tale dominio. Quando verifichi un dominio, consenti ad Amazon SES di inviare e-mail da qualsiasi indirizzo su quel dominio.

Quando esegui il seguente esempio di codice, Amazon SES ti fornisce un token di verifica. Devi aggiungere il token alla configurazione DNS del tuo dominio. Per ulteriori informazioni, consulta la

sezione Verifica di un dominio con Amazon SES nella Amazon Simple Email Service Developer Guide.

Per aggiungere un dominio di invio al tuo account Amazon SES, utilizza l'VerifyDomainIdentityoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca gli indirizzi e-mail

Per recuperare un elenco di indirizzi e-mail inviati nella AWS regione corrente, indipendentemente dallo stato di verifica, utilizza l'ListIdentitiesoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca i domini e-mail

Per recuperare un elenco di domini e-mail inviati nella AWS regione corrente, indipendentemente dallo stato di verifica, utilizza l'operazione. ListIdentities

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un indirizzo e-mail

Per eliminare un indirizzo email verificato dall'elenco delle identità, utilizza l'DeleteIdentityoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un dominio e-mail

Per eliminare un dominio e-mail verificato dall'elenco delle identità verificate, utilizza l'DeleteIdentityoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
```

```
}
```

Creazione di modelli di e-mail personalizzati utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3

Amazon Simple Email Service (Amazon SES) Simple Email Service (Amazon SES) consente di inviare e-mail personalizzate per ciascun destinatario utilizzando modelli. I modelli includono un oggetto, il testo e le parti HTML del corpo dell'e-mail. Le sezioni dell'oggetto e del corpo possono anche contenere valori univoci personalizzati per ogni destinatario.

Per ulteriori informazioni, consulta Invio di e-mail personalizzate utilizzando Amazon SES nella Amazon Simple Email Service Developer Guide.

Gli esempi seguenti mostrano come:

- Crea un modello di email utilizzando CreateTemplate.
- Elenca tutti i modelli di email che utilizzano ListTemplates.
- Recupera un modello di email utilizzando GetTemplate.
- Aggiorna un modello di email utilizzando UpdateTemplate.
- Rimuovi un modello di email utilizzando DeleteTemplate.
- Invia un'email basata su un modello utilizzando. SendTemplatedEmail

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Creazione di un modello di e-mail

Per creare un modello per inviare messaggi e-mail personalizzati, utilizza l'CreateTemplateoperazione. Il modello può essere utilizzato da qualsiasi account autorizzato a inviare messaggi nella AWS regione a cui è stato aggiunto il modello.

> **Note**
>
> Amazon SES non convalida il codice HTML, quindi assicurati che HtmlPartsia valido prima di inviare un'e-mail.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Richiedi un modello di email

Per visualizzare il contenuto di un modello di email esistente, inclusi l'oggetto, il corpo HTML e il testo semplice, utilizza l'GetTemplateoperazione. Solo TemplateName è obbligatorio.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca tutti i modelli di email

Per recuperare un elenco di tutti i modelli di email associati al tuo Account AWS nella AWS regione corrente, usa l'ListTemplatesoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Aggiornamento di un modello di e-mail

Per modificare il contenuto di un modello di email specifico, tra cui la riga dell'oggetto, il corpo HTML e il testo semplice, utilizza l'UpdateTemplateoperazione.

Importazioni

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

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

Codice di esempio

```
$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";
}
```

Eliminazione di un modello di e-mail

Per rimuovere un modello di email specifico, usa l'DeleteTemplateoperazione. Tutto ciò di cui hai bisogno è il TemplateName.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Inviare un'e-mail con un modello

Per utilizzare un modello per inviare un'e-mail ai destinatari, utilizza l'SendTemplatedEmailoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Gestione dei filtri e-mail tramite l'API Amazon SES e la AWS SDK per PHP versione 3

Oltre a inviare e-mail, puoi anche ricevere e-mail con Amazon Simple Email Service (Amazon SES). Un filtro degli indirizzi IP permette di specificare se accettare o rifiutare posta proveniente da un indirizzo IP o da un intervallo di indirizzi IP. Per ulteriori informazioni, consulta la pagina sulla gestione di filtri degli indirizzi IP per la ricezione di e-mail in Amazon SES.

Gli esempi seguenti mostrano come:

- Crea un filtro e-mail utilizzando CreateReceiptFilter.
- Elenca tutti i filtri e-mail utilizzando ListReceiptFilters.
- Rimuovi un filtro e-mail utilizzando DeleteReceiptFilter.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Crea un filtro e-mail

Per consentire o bloccare le e-mail provenienti da un indirizzo IP specifico, utilizza l'CreateReceiptFilteroperazione. Fornire l'indirizzo IP o l'intervallo di indirizzi e un nome univoco per identificare il filtro.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca tutti i filtri e-mail

Per elencare i filtri degli indirizzi IP associati alla tua Account AWS AWS regione nella regione corrente, usa l'ListReceiptFiltersoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un filtro e-mail

Per rimuovere un filtro esistente per un indirizzo IP specifico, utilizzare l'DeleteReceiptFilteroperazione. Fornire il nome di filtro univoco per identificare il filtro di ricezione da eliminare.

Se devi modificare l'intervallo di indirizzi filtrati, è possibile eliminare un filtro di ricezione e crearne uno nuovo.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Creazione e gestione di regole e-mail utilizzando l'API Amazon SES e la AWS SDK per PHP versione 3

Oltre a inviare e-mail, puoi anche ricevere e-mail con Amazon Simple Email Service (Amazon SES). Le regole di ricezione ti consentono di specificare cosa fa Amazon SES con le e-mail che riceve per gli indirizzi e-mail o i domini di tua proprietà. Una regola può inviare e-mail ad altri AWS servizi tra cui, a titolo esemplificativo ma non esaustivo, Amazon S3, Amazon SNS o. AWS Lambda

Per ulteriori informazioni, consulta Gestione dei set di regole di ricezione per Amazon SES Email Receiver e Gestione delle regole di ricezione per Amazon SES Email Reception.

Gli esempi seguenti mostrano come:

- Crea un set di regole di ricezione utilizzando CreateReceiptRuleSet.
- Crea una regola di incasso utilizzando CreateReceiptRule.
- Descrivi una regola di ricezione impostata utilizzando DescribeReceiptRuleSet.
- Descrivi una regola di incasso utilizzando DescribeReceiptRule.
- Elenca tutti i set di regole di ricezione utilizzando ListReceiptRuleSets.
- Aggiorna una regola di ricezione utilizzando UpdateReceiptRule.
- Rimuovi una regola di ricezione utilizzando DeleteReceiptRule.
- Rimuovi una regola di ricezione impostata utilizzando DeleteReceiptRuleSet.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Creazione di un set di regole di ricezione

Un set di regole di ricezione contiene una raccolta di regole di ricezione. È necessario avere almeno un set di regole di ricezione associato al tuo account prima di poter creare una regola di ricezione. Per creare un set di regole di ricezione, fornisci un'operazione univoca RuleSetName e utilizza l'CreateReceiptRuleSetoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Creazione di una regola di ricezione

Controlla la tua e-mail in entrata aggiungendo una regola di ricezione a un set di regole di ricezione esistente. Questo esempio mostra come creare una regola di ricezione che invii i messaggi in arrivo a un bucket Amazon S3, ma puoi anche inviare messaggi ad Amazon SNS e. AWS Lambda Per creare una regola di ricezione, fornisci una regola e il comando all' RuleSetName operazione. CreateReceiptRule

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Descrizione di un set di regole di ricezione

Una volta al secondo, restituisci i dettagli del set di regole di ricezione specificato. Per utilizzare l'DescribeReceiptRuleSetoperazione, fornisci il RuleSetName.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Descrivi una regola di ricezione

Restituire i dettagli di una regola di ricezione specifica. Per utilizzare l'DescribeReceiptRuleoperazione, fornisci il comando RuleName e RuleSetName.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Elenca tutti i set di regole di ricezione

Per elencare i set di regole di ricezione esistenti Account AWS nella AWS regione corrente, utilizza l'ListReceiptRuleSetsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Aggiornare una regola di incasso

Questo esempio mostra come aggiornare una regola di ricezione che invia i messaggi in entrata a una AWS Lambda funzione, ma puoi anche inviare messaggi ad Amazon SNS e Amazon S3. Per utilizzare l'UpdateReceiptRuleoperazione, fornisci la nuova regola di ricezione e il. RuleSetName

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminare un set di regole di incasso

Rimuovere un set di regole di ricezione specifico che non è attualmente disabilitato. Ciò elimina anche tutte le regole di ricezione che contiene. Per eliminare un set di regole di ricezione, fornisci il DeleteReceiptRuleSetcomando RuleSetName all'operazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Eliminazione di una regola di ricezione

Per eliminare una regola di ricezione specificata, fornite il comando RuleName e RuleSetName all'DeleteReceiptRuleoperazione.

Importazioni

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

use Aws\Exception\AwsException;
```

Codice di esempio

```
$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";
}
```

Monitoraggio dell'attività di invio tramite l'API Amazon SES e la AWS SDK per PHP versione 3

Amazon Simple Email Service (Amazon SES) fornisce metodi per monitorare l'attività di invio. Consigliamo di implementare questi metodi per tenere traccia di misure importanti, come le percentuali di mancati recapiti (bounce), reclami e messaggi rifiutati dell'account. Percentuali di rimbalzi e reclami eccessivamente elevate possono compromettere la tua capacità di inviare e-mail con Amazon SES.

Gli esempi seguenti mostrano come:

- Verifica la tua quota di invio utilizzando. GetSendQuota
- Monitora la tua attività di invio utilizzando GetSendStatistics.

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui GitHub.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Controlla la tua quota di invio

È possibile inviare solo una determinata quantità di messaggi in un singolo periodo di 24 ore. Per verificare quanti messaggi puoi ancora inviare, usa l'GetSendQuotaoperazione. Per ulteriori informazioni, consulta Gestire i limiti di invio di Amazon SES.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Monitora la tua attività di invio

Per recuperare le metriche dei messaggi che hai inviato nelle ultime due settimane, utilizza l'GetSendStatisticsoperazione. In questo esempio viene restituito il numero di tentativi di distribuzione, i mancati recapiti, i reclami e i messaggi rifiutati in incrementi di 15 minuti.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Autorizzazione dei mittenti tramite l'API Amazon SES e la versione 3 AWS SDK per PHP

Per consentire a un altro Account AWS AWS Identity and Access Management utente o AWS servizio di inviare e-mail tramite Amazon Simple Email Service (Amazon SES) Simple Email Service (Amazon SES) per tuo conto, devi creare una politica di autorizzazione all'invio. Questo è un documento in formato JSON che viene associato a un'identità di proprietà dell'utente.

La policy elenca in modo esplicito gli utenti a cui permetti di inviare e-mail per l'identità e in quali condizioni. A tutti i mittenti, eccetto te e le entità a cui concedi le autorizzazioni in modo esplicito nelle policy, non verrà concesso di inviare e-mail. A un'identità possono essere collegate nessuna, una o più policy. Puoi anche definire un'unica policy con più istruzioni per ottenere l'effetto di più policy.

Per ulteriori informazioni, consulta Utilizzo delle autorizzazioni all'invio con Amazon SES.

Gli esempi seguenti mostrano come:

- Crea un mittente autorizzato utilizzando PutIdentityPolicy.

- Recupera le politiche per un mittente autorizzato utilizzando. GetIdentityPolicies
- Elenca i mittenti autorizzati utilizzando. ListIdentityPolicies
- Revoca l'autorizzazione per un mittente autorizzato utilizzando. DeleteIdentityPolicy

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Per ulteriori informazioni sull'uso di Amazon SES, consulta la Amazon SES Developer Guide.

Crea un mittente autorizzato

Per autorizzare un altro utente Account AWS a inviare e-mail per tuo conto, utilizza una politica di identità per aggiungere o aggiornare l'autorizzazione all'invio di e-mail dai tuoi indirizzi e-mail o domini verificati. Per creare una politica di identità, usa l'PutIdentityPolicyoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Recupera le politiche per un mittente autorizzato

Restituisci le policy di autorizzazione all'invio associate a una specifica identità e-mail o a un'identità di dominio. Per ottenere l'autorizzazione all'invio per un determinato indirizzo email o dominio, usa l'operazione. GetIdentityPolicy

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Elenca i mittenti autorizzati

Per elencare le politiche di autorizzazione all'invio associate a un'identità di posta elettronica o a un'identità di dominio specifica AWS nella regione corrente, utilizza l'ListIdentityPoliciesoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Revoca l'autorizzazione per un mittente autorizzato

Rimuovi l'autorizzazione di invio Account AWS a un altro utente a inviare e-mail con un'identità di posta elettronica o un'identità di dominio eliminando la politica di identità associata all'operazione. DeleteIdentityPolicy

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Codice di esempio

```
$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";
}
```

Esempi di Amazon SNS con la versione 3 AWS SDK per PHP

Amazon Simple Notification Service (Amazon SNS) è un servizio web che coordina e gestisce la consegna o l'invio di messaggi agli endpoint o ai clienti abbonati.

In Amazon SNS esistono due tipi di client: editori (noti anche come produttori) e abbonati (detti anche consumatori). Gli editori comunicano in modo asincrono con i sottoscrittori producendo e inviando un messaggio a un argomento, che rappresenta un punto di accesso logico e un canale di comunicazione. Gli abbonati (server Web, indirizzi e-mail, code Amazon SQS AWS Lambda , funzioni) utilizzano o ricevono il messaggio o la notifica tramite uno dei protocolli supportati (Amazon SQS, URLs HTTP/HTTPS, e-mail AWS SMS, Lambda) quando sono abbonati all'argomento.

Tutto il codice di esempio per la versione 3 è disponibile qui. AWS SDK per PHP GitHub

Argomenti

- Gestione degli argomenti in Amazon SNS con la versione 3 AWS SDK per PHP
- Gestione degli abbonamenti in Amazon SNS AWS SDK per PHP con la versione 3
- Invio di messaggi SMS in Amazon SNS con la versione 3 AWS SDK per PHP

Gestione degli argomenti in Amazon SNS con la versione 3 AWS SDK per PHP

Per inviare notifiche ad Amazon Simple Queue Service (Amazon SQS), URLs HTTP/HTTPS, e-mail AWS Lambda o AWS SMS, devi prima creare un argomento che gestisca la consegna dei messaggi a tutti gli abbonati di quell'argomento.

In termini di modello di progettazione dell'osservatore, un argomento è come il soggetto. Dopo aver creato un argomento, è possibile aggiungere sottoscrittori che vengono automaticamente avvisati quando viene pubblicato un messaggio relativo all'argomento.

Scopri di più sulla sottoscrizione agli argomenti in Gestione degli abbonamenti in Amazon AWS SDK per PHP SNS con la versione 3.

Gli esempi seguenti mostrano come:

- Crea un argomento da utilizzare per la pubblicazione delle notifiche. CreateTopic
- Restituisce un elenco degli argomenti del richiedente utilizzando ListTopics.
- Elimina un argomento e tutte le relative sottoscrizioni utilizzando. DeleteTopic
- Restituisce tutte le proprietà di un argomento utilizzando GetTopicAttributes.
- Consenti al proprietario di un argomento di impostare un attributo dell'argomento su un nuovo valore utilizzando SetTopicAttributes.

Per ulteriori informazioni sull'uso di Amazon SNS, consulta Amazon SNS Topic Attributes for Message Delivery Status.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Creazione di un argomento

Per creare un argomento, usa l'CreateTopicoperazione.

Il nome di ogni argomento nel tuo Account AWS deve essere unico.

Importazioni

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

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Elenca i tuoi argomenti

Per elencare fino a 100 argomenti esistenti nella AWS regione corrente, usa l'ListTopicsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Eliminazione di un argomento

Per rimuovere un argomento esistente e tutte le relative sottoscrizioni, utilizzate l'DeleteTopicoperazione.

Tutti i messaggi non recapitati ancora agli abbonati verranno eliminati.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Ottieni gli attributi dell'argomento

Per recuperare le proprietà di un singolo argomento esistente, usa l'GetTopicAttributesoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Impostazione degli attributi degli argomenti

Per aggiornare le proprietà di un singolo argomento esistente, utilizzate l'SetTopicAttributesoperazione.

È possibile impostare solo gli attributi `Policy`, `DisplayName` e `DeliveryPolicy`.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Gestione degli abbonamenti in Amazon SNS AWS SDK per PHP con la versione 3

Utilizza gli argomenti di Amazon Simple Notification Service (Amazon SNS) per inviare notifiche ad Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, indirizzi e-mail, () oppure. AWS Server Migration Service AWS SMS AWS Lambda

Le sottoscrizioni sono collegate a un argomento che gestisce l'invio di messaggi ai sottoscrittori. Scopri di più sulla creazione di argomenti in Gestione degli argomenti in Amazon SNS con la AWS SDK per PHP versione 3.

Gli esempi seguenti mostrano come:

- Iscriviti a un argomento esistente utilizzando Subscribe.
- Verifica un abbonamento utilizzando ConfirmSubscription.
- Elenca gli abbonamenti esistenti utilizzando ListSubscriptionsByTopic.
- Elimina un abbonamento utilizzando Unsubscribe.
- Invia un messaggio a tutti i sottoscrittori di un argomento utilizzando Publish.

Per ulteriori informazioni sull'uso di Amazon SNS, consulta Using Amazon System-to-System SNS for Messaging.

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Sottoscrizione di un indirizzo e-mail a un argomento

Per avviare una sottoscrizione a un indirizzo e-mail, utilizzare l'operazione Subscribe.

Puoi utilizzare il metodo di sottoscrizione per sottoscrivere diversi endpoint a un argomento Amazon SNS, a seconda dei valori utilizzati per i parametri passati. Questo è illustrato in altri esempi di questo argomento.

In questo esempio, l'endpoint è un indirizzo e-mail. Un token di conferma viene inviato a questa e-mail. Verifica la sottoscrizione con questo token di conferma entro tre giorni dalla ricezione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Sottoscrivi un endpoint applicativo a un argomento

Per avviare una sottoscrizione a un'app Web, utilizza l'operazione Subscribe.

Puoi utilizzare il metodo di sottoscrizione per sottoscrivere diversi endpoint a un argomento Amazon SNS, a seconda dei valori utilizzati per i parametri passati. Questo è illustrato in altri esempi di questo argomento.

In questo esempio, l'endpoint è un URL. Un token di conferma viene inviato a questo indirizzo Web. Verifica la sottoscrizione con questo token di conferma entro tre giorni dalla ricezione.

Importazioni

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

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Sottoscrizione di una funzione Lambda a un argomento

Per avviare una sottoscrizione a una funzione Lambda, utilizzare l'operazione Subscribe.

Puoi utilizzare il metodo di sottoscrizione per sottoscrivere diversi endpoint a un argomento Amazon SNS, a seconda dei valori utilizzati per i parametri passati. Questo è illustrato in altri esempi di questo argomento.

In questo esempio, l'endpoint è una funzione Lambda. Un token di conferma viene inviato a questa funzione Lambda. Verifica la sottoscrizione con questo token di conferma entro tre giorni dalla ricezione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Sottoscrivi un SMS di testo a un argomento

Per inviare messaggi SMS a più numeri di telefono nello stesso momento, sottoscrivi ogni numero a un argomento.

Per avviare una sottoscrizione a un numero di telefono, utilizza l'operazione Subscribe.

Puoi utilizzare il metodo di sottoscrizione per sottoscrivere diversi endpoint a un argomento Amazon SNS, a seconda dei valori utilizzati per i parametri passati. Questo è illustrato in altri esempi di questo argomento.

In questo esempio, l'endpoint è un numero di telefono in formato E.164, uno standard per le telecomunicazioni internazionali.

Un token di conferma viene inviato a questo numero di telefono. Verifica la sottoscrizione con questo token di conferma entro tre giorni dalla ricezione.

Per un modo alternativo di inviare messaggi SMS con Amazon SNS, consulta Invio di messaggi SMS in Amazon SNS con AWS SDK per PHP la versione 3.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Conferma l'iscrizione a un argomento

Per creare una sottoscrizione, il proprietario dell'endpoint deve confermare l'intenzione di ricevere messaggi da un argomento utilizzando un token inviato quando una sottoscrizione viene inizialmente stabilita, come descritto in precedenza. I token di conferma sono validi per tre giorni. Dopo tre giorni, è possibile inviare nuovamente un token mediante la creazione di una nuova sottoscrizione.

Per confermare un abbonamento, usa l'ConfirmSubscriptionoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Elenca le sottoscrizioni a un argomento

Per elencare fino a 100 abbonamenti esistenti in una determinata AWS regione, usa l'ListSubscriptionsoperazione.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Annullamento della sottoscrizione a un argomento

Per rimuovere un endpoint da un argomento, utilizzare l'operazione Unsubscribe.

Se l'abbonamento richiede l'autenticazione per l'eliminazione, solo il proprietario dell'abbonamento o il proprietario dell'argomento può annullare l'iscrizione ed è richiesta una AWS firma. Se la chiamata di annullamento della registrazione non richiede l'autenticazione e il richiedente non è il proprietario della sottoscrizione, un messaggio di annullamento finale viene inviato all'endpoint.

Importazioni

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

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Pubblicare un messaggio su un argomento di Amazon SNS

Per recapitare un messaggio a ciascun endpoint sottoscritto a un argomento di Amazon SNS, utilizza l'operazione di pubblicazione.

Crea un oggetto che contenga i parametri per la pubblicazione di un messaggio, incluso il testo del messaggio e l'Amazon Resource Name (ARN) dell'argomento Amazon SNS.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Invio di messaggi SMS in Amazon SNS con la versione 3 AWS SDK per PHP

Puoi utilizzare Amazon Simple Notification Service (Amazon SNS) per inviare messaggi di testo o SMS a dispositivi dotati di SMS. Puoi inviare un messaggio direttamente a un numero di telefono oppure inviarlo a più numeri contemporaneamente sottoscrivendo quei numeri a un argomento e inviando il messaggio all'argomento.

Usa Amazon SNS per specificare le preferenze per la messaggistica SMS, ad esempio il modo in cui le consegne sono ottimizzate (in termini di costi o per una consegna affidabile), il limite di spesa mensile, il modo in cui vengono registrate le consegne dei messaggi e se abbonarsi ai report giornalieri sull'utilizzo degli SMS. Queste preferenze vengono recuperate e impostate come attributi SMS per Amazon SNS.

Quando invii un SMS, ricorda di specificare il numero di telefono utilizzando il formato E.164. E.164 è uno standard per la struttura del numero di telefono utilizzato per le telecomunicazioni internazionali. I numeri di telefono che seguono questo formato possono avere un massimo di 15 cifre e sono preceduti dal segno più (+) e dal prefisso del paese. Ad esempio, un numero di telefono statunitense in formato E.164 apparirebbe come +1001 0100. XXX555

Gli esempi seguenti mostrano come:

- Recupera le impostazioni predefinite per l'invio di messaggi SMS dal tuo account utilizzando GetSMSAttributes
- Aggiorna le impostazioni predefinite per l'invio di messaggi SMS dal tuo account utilizzando SetSMSAttributes.
- Scopri se il proprietario di un determinato numero di telefono ha scelto di non ricevere SMS dal tuo account utilizzando CheckIfPhoneNumberISOptedOut.
- Elenca i numeri di telefono utilizzati dal proprietario per i quali il proprietario ha scelto di non ricevere SMS dal tuo account. ListPhoneNumberOptedOut
- Invia un messaggio di testo SMS (messaggio) direttamente a un numero di telefono utilizzando Publish.

Per ulteriori informazioni sull'uso di Amazon SNS, consulta Utilizzo di Amazon SNS per le notifiche utente con un numero di cellulare come abbonato (invio SMS).

Tutto il codice di esempio per AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Ottieni gli attributi SMS

Per recuperare le impostazioni predefinite per i messaggi SMS, utilizzate l'SMSAttributesoperazione Get.

Questo esempio si riferisce all'attributo `DefaultSMSType`. Questo attributo consente di controllare se i messaggi SMS vengono inviati come `Promotional` per ottimizzare il recapito dei messaggi e permettere di contenere i costi, oppure come `Transactional` per ottimizzare il recapito dei messaggi e ottenere la massima affidabilità.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Imposta gli attributi SMS

Per aggiornare le impostazioni predefinite per i messaggi SMS, utilizzare l'SMSAttributesoperazione Set.

Questo esempio imposta l'attributo `DefaultSMSType` su `Transactional`, ottimizzando il recapito dei messaggi per ottenere la massima affidabilità.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Verifica se un numero di telefono è stato disattivato

Per determinare se il proprietario di un determinato numero di telefono ha scelto di non ricevere messaggi SMS dal tuo account, utilizza l'CheckIfPhoneNumberIsOptedOutoperazione.

In questo esempio, il numero di telefono è in formato E.164, uno standard per le telecomunicazioni internazionali.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Elenca i numeri di telefono esclusi

Per recuperare un elenco di numeri di telefono a cui il proprietario ha scelto di non ricevere SMS dal tuo account, utilizza l'operazione. ListPhoneNumbersOptedOut

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Pubblica su un messaggio di testo (messaggio SMS)

Per distribuire un messaggio di testo (messaggio SMS) direttamente a un numero di telefono, utilizzare l'operazione Publish.

In questo esempio, il numero di telefono è in formato E.164, uno standard per le telecomunicazioni internazionali.

I messaggi SMS possono contenere fino a 140 byte. Le dimensioni massime di una singola pubblicazione SMS sono di 1.600 byte.

Per ulteriori informazioni sull'invio di messaggi SMS, vedi Invio di un messaggio SMS.

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Codice di esempio

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

Esempi di Amazon SQS con la AWS SDK per PHP versione 3

Amazon Simple Queue Service (SQS) è un servizio di accodamento dei messaggi veloce, affidabile, scalabile e completamente gestito. Amazon SQS consente di disaccoppiare i componenti di un'applicazione cloud. Amazon SQS include code standard con throughput ed at-least-once

elaborazione elevati e code FIFO che forniscono la distribuzione FIFO (first>-in, first>-out) e l'elaborazione in una sola volta.

Tutto il codice di esempio per la versione 3 è disponibile qui. AWS SDK per PHP GitHub

Argomenti

- Abilitazione del polling lungo in Amazon SQS AWS SDK per PHP con la versione 3
- Gestione del timeout di visibilità in Amazon SQS AWS SDK per PHP con la versione 3
- Invio e ricezione di messaggi in Amazon SQS con AWS SDK per PHP la versione 3
- Utilizzo di code di lettere non scritte in Amazon SQS con la versione 3 AWS SDK per PHP
- Utilizzo delle code in Amazon SQS AWS SDK per PHP con la versione 3

Abilitazione del polling lungo in Amazon SQS AWS SDK per PHP con la versione 3

Il polling prolungato riduce il numero di risposte vuote consentendo ad Amazon SQS di attendere un periodo di tempo specificato affinché un messaggio diventi disponibile nella coda prima di inviare una risposta. Inoltre, il polling lungo elimina le risposte vuote false creando delle query per tutti i server invece di effettuarne il campionamento. Per abilitare il polling lungo, specifica un tempo di attesa diverso da zero per i messaggi ricevuti. Per ulteriori informazioni, consulta la sezione relativa al polling lungo di SQS.

Gli esempi seguenti mostrano come:

- Imposta gli attributi su una coda Amazon SQS per abilitare il polling lungo, utilizzando. SetQueueAttributes
- Recupera uno o più messaggi utilizzando long polling. ReceiveMessage
- Crea una lunga coda di sondaggi utilizzando. CreateQueue

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Imposta gli attributi su una coda per abilitare il polling lungo

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Recupera messaggi con sondaggi lunghi

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Crea una coda con sondaggi lunghi

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Gestione del timeout di visibilità in Amazon SQS AWS SDK per PHP con la versione 3

Un timeout di visibilità è un periodo di tempo durante il quale Amazon SQS impedisce ad altri componenti che consumano di ricevere ed elaborare un messaggio. Per ulteriori informazioni, consulta Timeout visibilità.

Gli esempi seguenti mostrano come:

- Modifica il timeout di visibilità di messaggi specificati in una coda con nuovi valori, utilizzando. ChangeMessageVisibilityBatch

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Modificare il timeout di visibilità di più messaggi

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Invio e ricezione di messaggi in Amazon SQS con AWS SDK per PHP la versione 3

Per ulteriori informazioni sui messaggi Amazon SQS, consulta Invio di un messaggio a una coda SQS e Ricezione ed eliminazione di un messaggio da una coda SQS nella Guida per l'utente di Service Quotas.

Gli esempi seguenti mostrano come:

- Invia un messaggio a una coda specificata utilizzando. SendMessage
- Recupera uno o più messaggi (fino a 10) da una coda specificata utilizzando. ReceiveMessage
- Elimina un messaggio da una coda utilizzando. DeleteMessage

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Invio di un messaggio

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Ricevere ed eliminare messaggi

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Utilizzo di code di lettere non scritte in Amazon SQS con la versione 3 AWS SDK per PHP

Una coda DLQ è una coda a cui altre code (origini) possono mirare per i messaggi che non possono essere elaborati correttamente. Puoi riservare e isolare questi messaggi nella coda DLQ per determinare perché l'elaborazione non è riuscita. Devi configurare singolarmente ogni coda di origine che invia messaggi a una coda DLQ. Code multiple possono mirare a una singola coda DLQ.

Per ulteriori informazioni, consulta Utilizzare le code DLQ di SQS.

Gli esempi seguenti mostrano come:

- Abilita una coda di lettere non scritte utilizzando. SetQueueAttributes

Tutto il codice di esempio per il AWS SDK per PHP è disponibile qui. GitHub

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Abilita una coda di lettere non scritte

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Utilizzo delle code in Amazon SQS AWS SDK per PHP con la versione 3

Per ulteriori informazioni sulle code Amazon SQS, consulta How SQS Queues Work.

Gli esempi seguenti mostrano come:

- Restituisci un elenco delle tue code utilizzando. ListQueues
- Crea una nuova coda utilizzando. CreateQueue
- Restituisce l'URL di una coda esistente utilizzando. GetQueueUrl
- Elimina una coda specificata utilizzando. DeleteQueue

Tutto il codice di esempio per il AWS SDK per PHP è disponibile GitHub qui.

Credenziali

Prima di eseguire il codice di esempio, configurate AWS le vostre credenziali, come descritto inCredenziali. Quindi importate il file AWS SDK per PHP, come descritto inUtilizzo di base.

Restituisce un elenco di code

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Crea una coda

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Restituisce l'URL di una coda

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Elimina una coda

Importazioni

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Codice di esempio

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

Invia eventi agli endpoint EventBridge globali di Amazon

Puoi utilizzare gli endpoint EventBridge globali di Amazon per migliorare la disponibilità e l'affidabilità delle tue applicazioni basate sugli eventi.

Dopo aver configurato l'endpoint EventBridge globale, puoi inviargli eventi utilizzando l'SDK for PHP.

> **Important**
>
> Per utilizzare gli endpoint EventBridge globali con l'SDK for PHP, nell'ambiente PHP deve essere installata l'estensione Common Runtime AWS (AWS CRT).

L'esempio seguente utilizza il PutEventsmetodo di inviare un singolo evento `EventBridgeClient` a un endpoint globale. EventBridge

```
<?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]
]);
```

Questo post del blog contiene ulteriori informazioni sugli endpoint EventBridge globali.

Esempi di codice SDK for PHP

Gli esempi di codice riportati in questo argomento mostrano come utilizzare AWS SDK per PHP with AWS.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Alcuni servizi contengono ulteriori categorie di esempi che mostrano come sfruttare le librerie o le funzioni specifiche del servizio.

Servizi

- Esempi di API Gateway con SDK for PHP
- Esempi di Aurora con SDK for PHP
- Esempi di Auto Scaling con SDK for PHP
- Esempi di Amazon Bedrock con SDK for PHP
- Esempi di Amazon Bedrock Runtime con SDK for PHP
- Esempi di Amazon DocumentDB con SDK for PHP
- Esempi di DynamoDB con SDK for PHP
- EC2 Esempi di Amazon che utilizzano SDK for PHP
- AWS Glue esempi che utilizzano SDK for PHP
- Esempi IAM che utilizzano SDK for PHP
- Esempi di Kinesis con SDK for PHP
- AWS KMS esempi che utilizzano SDK for PHP
- Esempi di Lambda con SDK for PHP
- Esempi di Amazon MSK con SDK for PHP

- Esempi di Amazon RDS con SDK for PHP
- Esempi di Amazon RDS Data Service con SDK for PHP
- Esempi di Amazon Rekognition con SDK for PHP
- Esempi di Amazon S3 con SDK for PHP
- Esempi di S3 Directory Bucket che utilizzano SDK for PHP
- Esempi di Amazon SES con SDK for PHP
- Esempi di Amazon SNS con SDK for PHP
- Esempi di Amazon SQS con SDK for PHP

Esempi di API Gateway con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with API Gateway.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

Azioni

`GetBasePathMapping`

Il seguente esempio di codice mostra come utilizzare`GetBasePathMapping`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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();
```

- Per i dettagli sull'API, consulta la GetBasePathMappingsezione AWS SDK per PHP API Reference.

`ListBasePathMappings`

Il seguente esempio di codice mostra come utilizzare`ListBasePathMappings`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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();
```

- Per i dettagli sull'API, consulta la ListBasePathMappingssezione AWS SDK per PHP API Reference.

`UpdateBasePathMapping`

Il seguente esempio di codice mostra come utilizzare`UpdateBasePathMapping`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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();
```

- Per i dettagli sull'API, consulta la UpdateBasePathMappingsezione AWS SDK per PHP API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Esempi di Aurora con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Aurora.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Esempi di Auto Scaling con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Auto Scaling.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao Auto Scaling

I seguenti esempi di codice mostrano come iniziare a usare Auto Scaling.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- Per i dettagli sull'API, consulta la DescribeAutoScalingGroupssezione AWS SDK per PHP API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un gruppo Amazon EC2 Auto Scaling con un modello di lancio e zone di disponibilità e ottieni informazioni sulle istanze in esecuzione.
- Abilita la raccolta di CloudWatch metriche Amazon.
- Aggiorna la capacità desiderata del gruppo e attendi l'avvio di un'istanza.
- Termina un'istanza nel gruppo.
- Elenca le attività di scalabilità che si verificano in risposta alle richieste degli utenti e ai cambiamenti di capacità.
- Ottieni statistiche per le CloudWatch metriche, quindi ripulisci le risorse.

SDK per PHP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Azioni

`CreateAutoScalingGroup`

Il seguente esempio di codice mostra come usare`CreateAutoScalingGroup`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- Per i dettagli sull'API, consulta la CreateAutoScalingGroupsezione AWS SDK per PHP API Reference.

`DeleteAutoScalingGroup`

Il seguente esempio di codice mostra come utilizzare`DeleteAutoScalingGroup`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Per i dettagli sull'API, consulta la DeleteAutoScalingGroupsezione AWS SDK per PHP API Reference.

`DescribeAutoScalingGroups`

Il seguente esempio di codice mostra come utilizzare`DescribeAutoScalingGroups`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- Per i dettagli sull'API, consulta la DescribeAutoScalingGroupssezione AWS SDK per PHP API Reference.

`DescribeAutoScalingInstances`

Il seguente esempio di codice mostra come utilizzare`DescribeAutoScalingInstances`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Per i dettagli sull'API, consulta la DescribeAutoScalingInstancessezione AWS SDK per PHP API Reference.

`DescribeScalingActivities`

Il seguente esempio di codice mostra come utilizzare`DescribeScalingActivities`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Per i dettagli sull'API, consulta la DescribeScalingActivitiessezione AWS SDK per PHP API Reference.

`DisableMetricsCollection`

Il seguente esempio di codice mostra come utilizzare`DisableMetricsCollection`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Per i dettagli sull'API, consulta la DisableMetricsCollectionsezione AWS SDK per PHP API Reference.

`EnableMetricsCollection`

Il seguente esempio di codice mostra come utilizzare`EnableMetricsCollection`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
```

```
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- Per i dettagli sull'API, consulta la EnableMetricsCollectionsezione AWS SDK per PHP API Reference.

`SetDesiredCapacity`

Il seguente esempio di codice mostra come utilizzare`SetDesiredCapacity`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- Per i dettagli sull'API, consulta la SetDesiredCapacitysezione AWS SDK per PHP API Reference.

`TerminateInstanceInAutoScalingGroup`

Il seguente esempio di codice mostra come utilizzare`TerminateInstanceInAutoScalingGroup`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
            }
        }
    }
```

- Per i dettagli sull'API, consulta la TerminateInstanceInAutoScalingGroupsezione AWS SDK per PHP API Reference.

`UpdateAutoScalingGroup`

Il seguente esempio di codice mostra come utilizzare`UpdateAutoScalingGroup`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
}
```

- Per i dettagli sull'API, consulta la UpdateAutoScalingGroupsezione AWS SDK per PHP API Reference.

Esempi di Amazon Bedrock con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Bedrock. AWS SDK per PHP

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

`ListFoundationModels`

Il seguente esempio di codice mostra come utilizzare`ListFoundationModels`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca i modelli Amazon Bedrock Foundation disponibili.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- Per i dettagli sulle API, consulta la sezione ListFoundationModels AWS SDK per PHPAPI Reference.

Esempi di Amazon Bedrock Runtime con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with Amazon Bedrock Runtime.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari
- Amazon Nova
- Amazon Titan Image Generator
- Anthropic Claude
- Diffusione stabile

Scenari

Richiama più modelli di base su Amazon Bedrock

Il seguente esempio di codice mostra come preparare e inviare un prompt a una varietà di modelli in grandi lingue (LLMs) su Amazon Bedrock

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Richiama più elementi LLMs su 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-v1');
        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;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - InvokeModel
  - InvokeModelWithResponseStream

Amazon Nova

conversare

Il seguente esempio di codice mostra come inviare un messaggio di testo ad Amazon Nova, utilizzando l'API Converse di Bedrock.

SDK per PHP

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Invia un messaggio di testo ad Amazon Nova utilizzando l'API Converse di 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();
```

- Per i dettagli sull'API, consulta Converse in AWS SDK per PHP API Reference.

Amazon Titan Image Generator

InvokeModel

Il seguente esempio di codice mostra come richiamare Amazon Titan Image su Amazon Bedrock per generare un'immagine.

SDK per PHP

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea un'immagine con 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-v1';
            $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;
    }
```

- Per i dettagli sull'API, consulta la sezione InvokeModel AWS SDK per PHPAPI Reference.

Anthropic Claude

InvokeModel

Il seguente esempio di codice mostra come inviare un messaggio di testo a Anthropic Claude, utilizzando l'API Invoke Model.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Invoca il modello di base Anthropic Claude 2 per generare testo.

```
    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;
    }
```

- Per i dettagli sulle API, consulta la sezione API InvokeModelReference AWS SDK per PHP .

Diffusione stabile

InvokeModel

Il seguente esempio di codice mostra come richiamare Stability.ai Stable Diffusion XL su Amazon Bedrock per generare un'immagine.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea un'immagine con Stable Diffusion.

```
    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;
    }
```

- Per i dettagli sull'API, consulta la sezione InvokeModel AWS SDK per PHPAPI Reference.

Esempi di Amazon DocumentDB con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon DocumentDB. AWS SDK per PHP

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Richiamare una funzione Lambda da un trigger Amazon DocumentDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Amazon DocumentDB con Lambda tramite 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();
```

Esempi di DynamoDB con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con DynamoDB.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea una tabella in grado di contenere i dati del filmato.
- Inserisci, ottieni e aggiorna un singolo filmato nella tabella.

- Scrivi i dati del filmato nella tabella da un file JSON di esempio.
- Esegui una query sui filmati che sono stati rilasciati in un dato anno.
- Cerca i filmati che sono stati distribuiti in diversi anni.
- Elimina un filmato dalla tabella, quindi elimina la tabella.

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
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);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Azioni

`BatchExecuteStatement`

Il seguente esempio di codice mostra come usare`BatchExecuteStatement`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
                ],
            ],
        ]);
    }
```

- Per i dettagli sull'API, consulta la BatchExecuteStatementsezione AWS SDK per PHP API Reference.

`BatchWriteItem`

Il seguente esempio di codice mostra come utilizzare`BatchWriteItem`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
            }
        }
    }
```

- Per i dettagli sull'API, consulta la BatchWriteItemsezione AWS SDK per PHP API Reference.

`CreateTable`

Il seguente esempio di codice mostra come utilizzare`CreateTable`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Creare una tabella .

```
        $tableName = "ddb_demo_table_$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],
        ]);
    }
```

- Per i dettagli sull'API, consulta la CreateTablesezione AWS SDK per PHP API Reference.

`DeleteItem`

Il seguente esempio di codice mostra come utilizzare`DeleteItem`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, consulta la DeleteItemsezione AWS SDK per PHP API Reference.

`DeleteTable`

Il seguente esempio di codice mostra come utilizzare`DeleteTable`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
            ]);
        });
    }
```

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK per PHP API Reference.

`ExecuteStatement`

Il seguente esempio di codice mostra come utilizzare`ExecuteStatement`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, consulta la ExecuteStatementsezione AWS SDK per PHP API Reference.

`GetItem`

Il seguente esempio di codice mostra come utilizzare`GetItem`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, consulta la GetItemsezione AWS SDK per PHP API Reference.

`ListTables`

Il seguente esempio di codice mostra come utilizzare`ListTables`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
            'ExclusiveStartTableName' => $exclusiveStartTableName,
            'Limit' => $limit,
        ]);
    }
```

- Per i dettagli sull'API, consulta la ListTablessezione AWS SDK per PHP API Reference.

`PutItem`

Il seguente esempio di codice mostra come utilizzare`PutItem`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, consulta la PutItemsezione AWS SDK per PHP API Reference.

`Query`

Il seguente esempio di codice mostra come utilizzare`Query`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK per PHP .

`Scan`

Il seguente esempio di codice mostra come usare`Scan`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per informazioni dettagliate sulle API, consulta Scan nella Documentazione di riferimento per le API AWS SDK per PHP .

`UpdateItem`

Il seguente esempio di codice mostra come usare`UpdateItem`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
                ]
            ],
        ]);
    }
```

- Per i dettagli sull'API, consulta la UpdateItemsezione AWS SDK per PHP API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Esecuzione di una query su una tabella mediante batch di istruzioni PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un batch di elementi mediante più istruzioni SELECT.
- Aggiunta di un batch di articoli eseguendo più istruzioni INSERT.
- Aggiornamento di un batch di elementi mediante più istruzioni UPDATE.
- Eliminazione di un batch di elementi mediante più istruzioni DELETE.

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
                ],
            ],
        ]);
    }
```

- Per i dettagli sull'API, consulta la BatchExecuteStatementsezione AWS SDK per PHP API Reference.

Esecuzione di una query mediante PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un articolo eseguendo un'istruzione SELECT.
- Aggiunta di un elemento eseguendo un'istruzione INSERT.
- Aggiornamento di un elemento eseguendo un'istruzione UPDATE.
- Eliminazione di un elemento eseguendo un'istruzione DELETE.

SDK per PHP

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
```

```
        ]);
    }
```

- Per i dettagli sull'API, consulta la ExecuteStatementsezione AWS SDK per PHP API Reference.

Esempi serverless

Richiamare una funzione Lambda da un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento DynamoDB con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso DynamoDB. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di DynamoDB con Lambda tramite 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 Esempi di Amazon che utilizzano SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il AWS SDK per PHP con Amazon EC2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

`CreateVpc`

Il seguente esempio di codice mostra come utilizzare`CreateVpc`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta la CreateVpcsezione AWS SDK per PHP API Reference.

`CreateVpcEndpoint`

Il seguente esempio di codice mostra come utilizzare`CreateVpcEndpoint`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta la CreateVpcEndpointsezione AWS SDK per PHP API Reference.

`DeleteVpc`

Il seguente esempio di codice mostra come utilizzare`DeleteVpc`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteVpcsezione AWS SDK per PHP API Reference.

`DeleteVpcEndpoints`

Il seguente esempio di codice mostra come utilizzare`DeleteVpcEndpoints`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta la DeleteVpcEndpointssezione AWS SDK per PHP API Reference.

`DescribeRouteTables`

Il seguente esempio di codice mostra come utilizzare`DescribeRouteTables`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
    }
```

- Per i dettagli sull'API, consulta la DescribeRouteTablessezione AWS SDK per PHP API Reference.

AWS Glue esempi che utilizzano SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with AWS Glue.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un crawler che esegue la scansione di un bucket Amazon S3 pubblico e genera un database di metadati in formato CSV.
- Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.
- Crea un processo per estrarre i dati CSV dal bucket S3, trasformare i dati e caricare l'output in formato JSON in un altro bucket S3.
- Elenca le informazioni sulle esecuzioni dei processi, visualizza i dati trasformati e pulisci le risorse.

Per ulteriori informazioni, consulta Tutorial: Guida introduttiva a AWS Glue Studio.

SDK per PHP

> **Note**
>
> C'è altro da sapere GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Azioni

`CreateCrawler`

Il seguente esempio di codice mostra come usare`CreateCrawler`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
                        ]]
                ],
            ]);
        });
    }
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK per PHP API Reference.

`CreateJob`

Il seguente esempio di codice mostra come utilizzare`CreateJob`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK per PHP API Reference.

`DeleteCrawler`

Il seguente esempio di codice mostra come utilizzare`DeleteCrawler`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Per i dettagli sull'API, consulta la DeleteCrawlersezione AWS SDK per PHP API Reference.

`DeleteDatabase`

Il seguente esempio di codice mostra come utilizzare`DeleteDatabase`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
```

```
    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK per PHP API Reference.

`DeleteJob`

Il seguente esempio di codice mostra come utilizzare`DeleteJob`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Per i dettagli sull'API, DeleteJobconsulta AWS SDK per PHP API Reference.

`DeleteTable`

Il seguente esempio di codice mostra come utilizzare`DeleteTable`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, DeleteTableconsulta AWS SDK per PHP API Reference.

`GetCrawler`

Il seguente esempio di codice mostra come utilizzare`GetCrawler`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
            ]);
        });
    }
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK per PHP API Reference.

`GetDatabase`

Il seguente esempio di codice mostra come utilizzare`GetDatabase`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
            ]);
        });
    }
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK per PHP API Reference.

`GetJobRun`

Il seguente esempio di codice mostra come utilizzare`GetJobRun`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
   }
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK per PHP API Reference.

`GetJobRuns`

Il seguente esempio di codice mostra come utilizzare`GetJobRuns`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK per PHP API Reference.

`GetTables`

Il seguente esempio di codice mostra come utilizzare`GetTables`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Per i dettagli sull'API, GetTablesconsulta AWS SDK per PHP API Reference.

`ListJobs`

Il seguente esempio di codice mostra come utilizzare`ListJobs`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListJobsconsulta AWS SDK per PHP API Reference.

`StartCrawler`

Il seguente esempio di codice mostra come utilizzare`StartCrawler`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
```

```
    }
```

- Per i dettagli sull'API, StartCrawlerconsulta AWS SDK per PHP API Reference.

`StartJobRun`

Il seguente esempio di codice mostra come utilizzare`StartJobRun`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
           ],
       ]);
   }
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK per PHP API Reference.

Esempi IAM che utilizzano SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with IAM.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

Il seguente esempio di codice mostra come creare un utente e assumere un ruolo.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

- Crea un utente che non disponga di autorizzazioni.
- Crea un ruolo che conceda l'autorizzazione per elencare i bucket Amazon S3 per l'account.

- Aggiungi una policy per consentire all'utente di assumere il ruolo.
- Assumi il ruolo ed elenca i bucket S3 utilizzando le credenziali temporanee, quindi ripulisci le risorse.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole

- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Azioni

`AttachRolePolicy`

Il seguente esempio di codice mostra come usare`AttachRolePolicy`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
            ]);
        });
    }
```

- Per i dettagli sull'API, AttachRolePolicyconsulta AWS SDK per PHP API Reference.

`CreatePolicy`

Il seguente esempio di codice mostra come utilizzare`CreatePolicy`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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'];
    }
```

- Per i dettagli sull'API, CreatePolicyconsulta AWS SDK per PHP API Reference.

`CreateRole`

Il seguente esempio di codice mostra come utilizzare`CreateRole`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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'];
    }
```

- Per i dettagli sull'API, CreateRoleconsulta AWS SDK per PHP API Reference.

`CreateServiceLinkedRole`

Il seguente esempio di codice mostra come utilizzare`CreateServiceLinkedRole`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, CreateServiceLinkedRoleconsulta AWS SDK per PHP API Reference.

`CreateUser`

Il seguente esempio di codice mostra come utilizzare`CreateUser`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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'];
    }
```

- Per i dettagli sull'API, CreateUserconsulta AWS SDK per PHP API Reference.

`GetAccountPasswordPolicy`

Il seguente esempio di codice mostra come utilizzare`GetAccountPasswordPolicy`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Per i dettagli sull'API, GetAccountPasswordPolicyconsulta AWS SDK per PHP API Reference.

`GetPolicy`

Il seguente esempio di codice mostra come utilizzare`GetPolicy`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- Per i dettagli sull'API, GetPolicyconsulta AWS SDK per PHP API Reference.

`GetRole`

Il seguente esempio di codice mostra come utilizzare`GetRole`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
```

```
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Per i dettagli sull'API, GetRoleconsulta AWS SDK per PHP API Reference.

`ListAttachedRolePolicies`

Il seguente esempio di codice mostra come utilizzare`ListAttachedRolePolicies`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListAttachedRolePoliciesconsulta AWS SDK per PHP API Reference.

`ListGroups`

Il seguente esempio di codice mostra come utilizzare`ListGroups`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListGroupsconsulta AWS SDK per PHP API Reference.

`ListPolicies`

Il seguente esempio di codice mostra come utilizzare`ListPolicies`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListPoliciesconsulta AWS SDK per PHP API Reference.

`ListRolePolicies`

Il seguente esempio di codice mostra come utilizzare`ListRolePolicies`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
        });
    }
```

- Per i dettagli sull'API, ListRolePoliciesconsulta AWS SDK per PHP API Reference.

`ListRoles`

Il seguente esempio di codice mostra come utilizzare`ListRoles`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListRolesconsulta AWS SDK per PHP API Reference.

`ListSAMLProviders`

Il seguente esempio di codice mostra come utilizzare`ListSAMLProviders`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Per i dettagli sull'API, consulta List SAMLProviders in AWS SDK per PHP API Reference.

`ListUsers`

Il seguente esempio di codice mostra come utilizzare`ListUsers`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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);
    }
```

- Per i dettagli sull'API, ListUsersconsulta AWS SDK per PHP API Reference.

Esempi di Kinesis con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Kinesis.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Kinesis con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di Kinesis con Lambda tramite 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 esempi che utilizzano SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with AWS KMS.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Ciao AWS Key Management Service

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Key Management Service.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
}
```

- Per i dettagli sull'API, ListKeysconsulta AWS SDK per PHP API Reference.

Argomenti

- Nozioni di base
- Azioni

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Creare una chiave KMS.
- Elenca le chiavi KMS del tuo account e ottieni dettagli su di esse.
- Abilitare e disabilitare le chiavi KMS.
- Genera una chiave dati simmetrica che può essere utilizzata per la crittografia lato client.
- Genera una chiave asimmetrica utilizzata per firmare digitalmente i dati.
- Chiavi per tag.
- Eliminare le chiavi KMS.

SDK per PHP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }


}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign

- TagResource

Azioni

`CreateAlias`

Il seguente esempio di codice mostra come usare`CreateAlias`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, CreateAliasconsulta AWS SDK per PHP API Reference.

`CreateGrant`

Il seguente esempio di codice mostra come utilizzare`CreateGrant`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, CreateGrantconsulta AWS SDK per PHP API Reference.

`CreateKey`

Il seguente esempio di codice mostra come utilizzare`CreateKey`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
```

```
        }
    }
```

- Per i dettagli sull'API, CreateKeyconsulta AWS SDK per PHP API Reference.

`Decrypt`

Il seguente esempio di codice mostra come utilizzare`Decrypt`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta Decrypt in AWS SDK per PHP API Reference.

`DeleteAlias`

Il seguente esempio di codice mostra come utilizzare. `DeleteAlias`

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, DeleteAliasconsulta AWS SDK per PHP API Reference.

`DescribeKey`

Il seguente esempio di codice mostra come utilizzare`DescribeKey`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, DescribeKeyconsulta AWS SDK per PHP API Reference.

`DisableKey`

Il seguente esempio di codice mostra come utilizzare`DisableKey`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, DisableKeyconsulta AWS SDK per PHP API Reference.

`EnableKey`

Il seguente esempio di codice mostra come utilizzare`EnableKey`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, EnableKeyconsulta AWS SDK per PHP API Reference.

`Encrypt`

Il seguente esempio di codice mostra come utilizzare`Encrypt`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta Encrypt in AWS SDK per PHP API Reference.

`ListAliases`

Il seguente esempio di codice mostra come utilizzare`ListAliases`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, ListAliasesconsulta AWS SDK per PHP API Reference.

`ListGrants`

Il seguente esempio di codice mostra come utilizzare`ListGrants`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, ListGrantsconsulta AWS SDK per PHP API Reference.

`ListKeys`

Il seguente esempio di codice mostra come utilizzare`ListKeys`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, ListKeysconsulta AWS SDK per PHP API Reference.

`PutKeyPolicy`

Il seguente esempio di codice mostra come utilizzare`PutKeyPolicy`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
```

```
        }
    }
```

- Per i dettagli sull'API, PutKeyPolicyconsulta AWS SDK per PHP API Reference.

`RevokeGrant`

Il seguente esempio di codice mostra come utilizzare`RevokeGrant`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, RevokeGrantconsulta AWS SDK per PHP API Reference.

`ScheduleKeyDeletion`

Il seguente esempio di codice mostra come utilizzare`ScheduleKeyDeletion`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, ScheduleKeyDeletionconsulta AWS SDK per PHP API Reference.

`Sign`

Il seguente esempio di codice mostra come utilizzare`Sign`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, consulta Sign in AWS SDK per PHP API Reference.

`TagResource`

Il seguente esempio di codice mostra come usare`TagResource`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, TagResourceconsulta AWS SDK per PHP API Reference.

Esempi di Lambda con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with Lambda.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, in cui è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un ruolo IAM e una funzione Lambda, quindi carica il codice del gestore.
- Richiamare la funzione con un singolo parametro e ottenere i risultati.
- Aggiorna il codice della funzione e configuralo con una variabile di ambiente.
- Richiamare la funzione con nuovi parametri e ottenere i risultati. Visualizza il log di esecuzione restituito.
- Elenca le funzioni dell'account, quindi elimina le risorse.

Per ulteriori informazioni sull'utilizzo di Lambda, consulta Creare una funzione Lambda con la console.

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
        $bucketName = "test-example-bucket-$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";
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Azioni

`CreateFunction`

Il seguente esempio di codice mostra come usare`CreateFunction`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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",
            ]);
        });
    }
```

- Per i dettagli sull'API, CreateFunctionconsulta AWS SDK per PHP API Reference.

`DeleteFunction`

Il seguente esempio di codice mostra come utilizzare`DeleteFunction`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Per i dettagli sull'API, DeleteFunctionconsulta AWS SDK per PHP API Reference.

`GetFunction`

Il seguente esempio di codice mostra come utilizzare`GetFunction`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Per i dettagli sull'API, GetFunctionconsulta AWS SDK per PHP API Reference.

`Invoke`

Il seguente esempio di codice mostra come utilizzare`Invoke`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- Per informazioni dettagliate sulle API, consulta Invoke nella Documentazione di riferimento delle API AWS SDK per PHP .

`ListFunctions`

Il seguente esempio di codice mostra come usare`ListFunctions`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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,
        ]);
    }
```

- Per i dettagli sull'API, ListFunctionsconsulta AWS SDK per PHP API Reference.

`UpdateFunctionCode`

Il seguente esempio di codice mostra come utilizzare`UpdateFunctionCode`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- Per i dettagli sull'API, UpdateFunctionCodeconsulta AWS SDK per PHP API Reference.

`UpdateFunctionConfiguration`

Il seguente esempio di codice mostra come utilizzare`UpdateFunctionConfiguration`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- Per i dettagli sull'API, UpdateFunctionConfigurationconsulta AWS SDK per PHP API Reference.

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Esempi serverless

Connessione a un database Amazon RDS in una funzione Lambda

Il seguente esempio di codice mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Connessione a un database Amazon RDS in una funzione Lambda tramite 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);
```

Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Kinesis con Lambda tramite 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);
```

Richiamare una funzione Lambda da un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento DynamoDB con Lambda tramite 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);
```

Richiamare una funzione Lambda da un trigger Amazon DocumentDB

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Amazon DocumentDB con Lambda tramite 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();
```

Invocare una funzione Lambda da un trigger Amazon MSK

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento generato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Amazon MSK con Lambda tramite 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);
```

Richiamo di una funzione Lambda da un trigger Amazon S3

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento S3 con Lambda tramite 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);
```

Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SNS con Lambda tramite 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();
```

Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SQS con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di Kinesis con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger DynamoDB

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso DynamoDB. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di DynamoDB con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di SQS con Lambda tramite 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);
```

Esempi di Amazon MSK con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Amazon MSK.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Invocare una funzione Lambda da un trigger Amazon MSK

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento generato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK e registra il contenuto del record.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento Amazon MSK con Lambda tramite 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);
```

Esempi di Amazon RDS con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon RDS. AWS SDK per PHP

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari
- Esempi serverless

Azioni

`CreateDBInstance`

Il seguente esempio di codice mostra come utilizzare`CreateDBInstance`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
}
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per PHP API Reference.

`CreateDBSnapshot`

Il seguente esempio di codice mostra come utilizzare`CreateDBSnapshot`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
}
```

- Per i dettagli sull'API, consulta Create DBSnapshot in AWS SDK per PHP API Reference.

`DeleteDBInstance`

Il seguente esempio di codice mostra come utilizzare`DeleteDBInstance`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per PHP API Reference.

`DescribeDBInstances`

Il seguente esempio di codice mostra come utilizzare`DescribeDBInstances`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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";
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per PHP API Reference.

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Esempi serverless

Connessione a un database Amazon RDS in una funzione Lambda

Il seguente esempio di codice mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Connessione a un database Amazon RDS in una funzione Lambda tramite 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);
```

Esempi di Amazon RDS Data Service con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Amazon RDS Data Service.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Esempi di Amazon Rekognition con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon AWS SDK per PHP Rekognition.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

> Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.
>
> Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.
>
> Per approfondire l'origine di questo esempio, consulta il post su AWS Community.
>
> Servizi utilizzati in questo esempio
>
> - API Gateway
> - DynamoDB
> - Lambda
> - Amazon Rekognition
> - Amazon S3
> - Amazon SNS

Esempi di Amazon S3 con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon S3. AWS SDK per PHP

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Nozioni di base

Hello Amazon S3

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon S3.

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
use Aws\S3\S3Client;

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

- Per i dettagli sull'API, ListBucketsconsulta AWS SDK per PHP API Reference.

Argomenti

- Nozioni di base
- Azioni
- Scenari
- Esempi serverless

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un bucket e carica un file in tale bucket.
- Scaricare un oggetto da un bucket.
- Copiare un oggetto in una sottocartella in un bucket.
- Elencare gli oggetti in un bucket.
- Elimina il bucket e tutti gli oggetti in esso contenuti.

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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 (count($check) <= 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";
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Azioni

`CopyObject`

Il seguente esempio di codice mostra come usare`CopyObject`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Copia semplice di un oggetto.

```
        $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.");
        }
```

- Per i dettagli sull'API, CopyObjectconsulta AWS SDK per PHP API Reference.

`CreateBucket`

Il seguente esempio di codice mostra come utilizzare`CreateBucket`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Creare un 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.");
        }
```

- Per i dettagli sull'API, CreateBucketconsulta AWS SDK per PHP API Reference.

`DeleteBucket`

Il seguente esempio di codice mostra come utilizzare`DeleteBucket`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elimina un bucket vuoto.

```
        $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.");
        }
```

- Per i dettagli sull'API, DeleteBucketconsulta AWS SDK per PHP API Reference.

`DeleteObject`

Il seguente esempio di codice mostra come utilizzare`DeleteObject`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }
```

- Per i dettagli sull'API, DeleteObjectconsulta AWS SDK per PHP API Reference.

`DeleteObjects`

Il seguente esempio di codice mostra come utilizzare`DeleteObjects`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elimina un set di oggetti da un elenco di chiavi.

```
        $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' => $this->bucketName,
            ]);
            if (count($check) <= 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.");
        }
```

- Per i dettagli sull'API, DeleteObjectsconsulta AWS SDK per PHP API Reference.

`GetObject`

Il seguente esempio di codice mostra come utilizzare`GetObject`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Recupera un oggetto.

```
        $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.");
        }
```

- Per i dettagli sull'API, GetObjectconsulta AWS SDK per PHP API Reference.

`ListObjectsV2`

Il seguente esempio di codice mostra come utilizzare`ListObjectsV2`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca gli oggetti in un 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.");
        }
```

- Per i dettagli sull'API, consulta la ListObjectsversione V2 in AWS SDK per PHP API Reference.

`PutObject`

Il seguente esempio di codice mostra come utilizzare`PutObject`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Carica un oggetto in un 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.");
        }
```

- Per i dettagli sull'API, PutObjectconsulta AWS SDK per PHP API Reference.

Scenari

Creazione di un URL prefirmato

Il seguente esempio di codice mostra come creare un URL predefinito per Amazon S3 e caricare un oggetto.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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;
        }
    }

}
```

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon S3

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento S3 con Lambda tramite 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);
```

Esempi di S3 Directory Bucket che utilizzano SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP with S3 Directory Buckets.

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Nozioni di base

Nozioni di base

Informazioni di base

L'esempio di codice seguente mostra come:

- Configura un VPC e un endpoint VPC.
- Configura le politiche, i ruoli e l'utente per utilizzare i bucket di directory S3 e la classe di storage S3 Express One Zone.
- Crea due client S3.
- Creare due bucket.
- Crea un oggetto e copialo.
- Dimostra la differenza di prestazioni.
- Compila i bucket per mostrare la differenza lessicografica.
- Chiedi all'utente di vedere se desidera ripulire le risorse.

SDK per PHP

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario che dimostri le nozioni di base dei bucket di directory Amazon S3 e di 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;
        }
    }

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per PHP .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObject
- GetObject
- ListObjects
- PutObject

Esempi di Amazon SES con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Amazon SES.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Scenari

Scenari

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Il seguente esempio di codice mostra come creare un'applicazione Web che tiene traccia degli elementi di lavoro in un database Amazon Aurora Serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.

- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Esempi di Amazon SNS con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Amazon SNS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari
- Esempi serverless

Azioni

`CheckIfPhoneNumberIsOptedOut`

Il seguente esempio di codice mostra come utilizzare`CheckIfPhoneNumberIsOptedOut`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per i dettagli sull'API, CheckIfPhoneNumberIsOptedOutconsulta AWS SDK per PHP API Reference.

`ConfirmSubscription`

Il seguente esempio di codice mostra come utilizzare`ConfirmSubscription`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, ConfirmSubscriptionconsulta AWS SDK per PHP API Reference.

`CreateTopic`

Il seguente esempio di codice mostra come utilizzare`CreateTopic`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per i dettagli sull'API, CreateTopicconsulta AWS SDK per PHP API Reference.

`DeleteTopic`

Il seguente esempio di codice mostra come utilizzare`DeleteTopic`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, DeleteTopicconsulta AWS SDK per PHP API Reference.

`GetSMSAttributes`

Il seguente esempio di codice mostra come utilizzare`GetSMSAttributes`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per i dettagli sull'API, consulta Get SMSAttributes in AWS SDK per PHP API Reference.

`GetTopicAttributes`

Il seguente esempio di codice mostra come usare`GetTopicAttributes`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, GetTopicAttributesconsulta AWS SDK per PHP API Reference.

`ListPhoneNumbersOptedOut`

Il seguente esempio di codice mostra come utilizzare`ListPhoneNumbersOptedOut`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per i dettagli sull'API, ListPhoneNumbersOptedOutconsulta AWS SDK per PHP API Reference.

`ListSubscriptions`

Il seguente esempio di codice mostra come utilizzare`ListSubscriptions`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, ListSubscriptionsconsulta AWS SDK per PHP API Reference.

`ListTopics`

Il seguente esempio di codice mostra come utilizzare`ListTopics`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, ListTopicsconsulta AWS SDK per PHP API Reference.

`Publish`

Il seguente esempio di codice mostra come utilizzare`Publish`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK per PHP .

`SetSMSAttributes`

Il seguente esempio di codice mostra come usare`SetSMSAttributes`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per i dettagli sull'API, consulta Set SMSAttributes in AWS SDK per PHP API Reference.

`SetTopicAttributes`

Il seguente esempio di codice mostra come utilizzare`SetTopicAttributes`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per i dettagli sull'API, consulta la SetTopicAttributessezione AWS SDK per PHP API Reference.

`Subscribe`

Il seguente esempio di codice mostra come utilizzare`Subscribe`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Sottoscrivi un indirizzo email a un argomento.

```
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());
}
```

Sottoscrivi un endpoint HTTP a un argomento.

```
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());
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK per PHP .

`Unsubscribe`

Il seguente esempio di codice mostra come utilizzare`Unsubscribe`.

SDK per PHP

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK per PHP .

Scenari

Creazione di un'applicazione serverless per gestire foto

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Pubblicazione di un SMS

Il seguente esempio di codice mostra come pubblicare messaggi SMS utilizzando Amazon SNS.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK per PHP .

Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SNS con Lambda tramite 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();
```

Esempi di Amazon SQS con SDK for PHP

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per PHP con Amazon SQS.

Ogni esempio include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SQS con Lambda tramite 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);
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di SQS con Lambda tramite 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);
```

Sicurezza per AWS SDK per PHP

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In quanto cliente AWS , è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza. La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud.

Security of the Cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce tutti i servizi offerti nel AWS Cloud e della fornitura di servizi che è possibile utilizzare in modo sicuro. La nostra responsabilità in AWS materia di sicurezza è la massima priorità e l'efficacia della nostra sicurezza viene regolarmente testata e verificata da revisori di terze parti nell'ambito dei Programmi di AWS conformità.

Sicurezza nel cloud: la responsabilità dell'utente è determinata dal AWS servizio utilizzato e da altri fattori, tra cui la sensibilità dei dati, i requisiti dell'organizzazione e le leggi e i regolamenti applicabili.

Argomenti

- Protezione dei dati dell' AWS SDK per PHP
- Identity and Access Management
- Convalida della conformità per questo AWS prodotto o servizio
- Resilienza per questo AWS prodotto o servizio
- Sicurezza dell'infrastruttura per questo AWS prodotto o servizio
- Migrazione del client di crittografia Amazon S3

Protezione dei dati dell' AWS SDK per PHP

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori AWS SDK per PHP o Servizi AWS utilizzi la console, l'API o. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Identity and Access Management

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse. AWS IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari

- Autenticazione con identità
- Gestione dell'accesso con policy
- Come Servizi AWS lavorare con IAM
- Risoluzione dei problemi di AWS identità e accesso

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro che AWS svolgi.

Utente del servizio: se lo utilizzi Servizi AWS per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più AWS funzionalità per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AWS, consulta Risoluzione dei problemi di AWS identità e accesso o consulta la guida per l'utente della funzionalità Servizio AWS che stai utilizzando.

Amministratore del servizio: se sei responsabile delle AWS risorse della tua azienda, probabilmente hai pieno accesso a AWS. È tuo compito determinare a quali AWS funzionalità e risorse devono accedere gli utenti del servizio. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per saperne di più su come la tua azienda può utilizzare IAM con AWS, consulta la guida per l'utente del Servizio AWS software che stai utilizzando.

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS. Per visualizzare esempi di policy AWS basate sull'identità che puoi utilizzare in IAM, consulta la guida per l'utente di quella Servizio AWS che stai utilizzando.

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On

della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sul metodo consigliato per la firma delle richieste, consulta Signature Version 4 AWS per le richieste API nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA)AWS in IAM nella Guida per l'utente IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi

AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni su IAM Identity Center, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Per assumere temporaneamente un ruolo IAM in AWS Management Console, puoi passare da un ruolo utente a un ruolo IAM (console). Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Create a role for a third-party identity provider (federation) nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza e che AWS CLI effettuano richieste AWS API. È preferibile archiviare le chiavi di accesso all'interno dell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal`sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente IAM.
- Politiche di controllo del servizio (SCPs): SCPs sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più di proprietà dell' Account AWS azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità presenti negli account dei membri, inclusa ciascuna di esse. Utente root dell'account AWS Per ulteriori informazioni su Organizations and SCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- Politiche di controllo delle risorse (RCPs): RCPs sono politiche JSON che puoi utilizzare per impostare le autorizzazioni massime disponibili per le risorse nei tuoi account senza aggiornare le politiche IAM allegate a ciascuna risorsa di tua proprietà. L'RCP limita le autorizzazioni per le risorse negli account dei membri e può influire sulle autorizzazioni effettive per le identità, incluse le Utente root dell'account AWS, indipendentemente dal fatto che appartengano o meno all'organizzazione. Per ulteriori informazioni su Organizations e RCPs, incluso un elenco di Servizi AWS tale supporto RCPs, vedere Resource control policies (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta la logica di valutazione delle policy nella IAM User Guide.

Come Servizi AWS lavorare con IAM

Per avere una visione di alto livello di come Servizi AWS funziona la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Per scoprire come utilizzare uno specifico Servizio AWS con IAM, consulta la sezione sulla sicurezza della Guida per l'utente del servizio pertinente.

Risoluzione dei problemi di AWS identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con un AWS IAM.

Argomenti

- Non sono autorizzato a eseguire alcuna azione in AWS
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse

Non sono autorizzato a eseguire alcuna azione in AWS

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `awes:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `awes:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie AWS risorse

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se AWS supporta queste funzionalità, consulta. Come Servizi AWS lavorare con IAM
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.

- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Convalida della conformità per questo AWS prodotto o servizio

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Governance e conformità per la sicurezza: queste guide all'implementazione di soluzioni illustrano considerazioni relative all'architettura e i passaggi per implementare le funzionalità di sicurezza e conformità.
- Riferimenti sui servizi conformi ai requisiti HIPAA: elenca i servizi HIPAA idonei. Non tutti Servizi AWS sono idonei alla normativa HIPAA.
- AWS Risorse per la per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.
- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).

- Valutazione delle risorse con regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, come lo standard PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.
- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente l' AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso gli specifici servizi Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Resilienza per questo AWS prodotto o servizio

L'infrastruttura AWS globale è costruita attorno a zone Regioni AWS di disponibilità.

Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti.

Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, vedere Global Infrastructure.AWS

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso gli specifici servizi Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Sicurezza dell'infrastruttura per questo AWS prodotto o servizio

Questo AWS prodotto o servizio utilizza servizi gestiti ed è pertanto protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzate chiamate API AWS pubblicate per accedere a questo AWS Prodotto o Servizio attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Questo AWS prodotto o servizio segue il modello di responsabilità condivisa attraverso gli specifici servizi Amazon Web Services (AWS) che supporta. Per informazioni sulla sicurezza dei AWS servizi, consulta la pagina della documentazione sulla sicurezza del AWS servizio e AWS i servizi che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Migrazione del client di crittografia Amazon S3

Questo argomento mostra come migrare le applicazioni dalla versione 1 (V1) del client di crittografia Amazon Simple Storage Service (Amazon S3) alla versione 2 (V2) e garantire la disponibilità delle applicazioni durante tutto il processo di migrazione.

Panoramica della migrazione

Questa migrazione avviene in due fasi:

1. Aggiorna i client esistenti per leggere nuovi formati. Innanzitutto, distribuisci una versione aggiornata di nella AWS SDK per PHP tua applicazione. Ciò consente ai client di crittografia V1

esistenti di decrittografare gli oggetti scritti dai nuovi client V2. Se l'applicazione ne utilizza più AWS SDKs, è necessario aggiornare ogni SDK separatamente.

2. Migra i client di crittografia e decrittografia alla versione 2. Una volta che tutti i client di crittografia V1 sono in grado di leggere nuovi formati, è possibile migrare i client di crittografia e decrittografia esistenti alle rispettive versioni V2.

Aggiorna i client esistenti per leggere nuovi formati

Il client di crittografia V2 utilizza algoritmi di crittografia che le versioni precedenti del client non supportano. Il primo passo della migrazione consiste nell'aggiornare i client di decrittografia V1 all'ultima versione dell'SDK. Dopo aver completato questo passaggio, i client V1 dell'applicazione saranno in grado di decrittografare gli oggetti crittografati dai client di crittografia V2. Vedi i dettagli di seguito per ciascuna versione principale di. AWS SDK per PHP

Aggiornamento della versione 3 AWS SDK per PHP

La versione 3 è l'ultima versione di. AWS SDK per PHP Per completare questa migrazione, è necessario utilizzare la versione 3.148.0 o successiva del pacchetto. `aws/aws-sdk-php`

Installazione dalla riga di comando

Per i progetti che sono stati installati utilizzando Composer, nel file Composer, aggiorna il pacchetto SDK alla versione 3.148.0 dell'SDK, quindi esegui il comando seguente.

```
composer update aws/aws-sdk-php
```

Installazione mediante il file Phar o Zip

Utilizzare uno dei seguenti metodi. Assicurati di inserire il file SDK aggiornato nella posizione richiesta dal codice, determinata dall'istruzione require.

Per i progetti che sono stati installati utilizzando il file Phar, scaricate il file aggiornato:. `aws.phar`

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

Per i progetti che sono stati installati utilizzando il file Zip, scarica il file aggiornato: .

```
<?php
```

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

Migra i client di crittografia e decrittografia alla V2

Dopo aver aggiornato i client per leggere i nuovi formati di crittografia, è possibile aggiornare le applicazioni ai client di crittografia e decrittografia V2. I passaggi seguenti mostrano come migrare correttamente il codice dalla V1 alla V2.

Requisiti per l'aggiornamento ai client V2

1. Il contesto di AWS KMS crittografia deve essere passato ai `S3EncryptionClientV2::putObjectAsync` metodi `S3EncryptionClientV2::putObject` and. AWS KMS il contesto di crittografia è un array associativo di coppie chiave-valore, che è necessario aggiungere al contesto di crittografia per AWS KMS la crittografia a chiave. Se non è richiesto alcun contesto aggiuntivo, è possibile passare un array vuoto.

2. `@SecurityProfile`deve essere passato ai `getObjectAsync` metodi `getObject` e in`S3EncryptionClientV2`. `@SecurityProfile`è un nuovo parametro obbligatorio dei `getObject...` metodi. Se impostato su`'V2'`, solo gli oggetti crittografati in un formato compatibile con V2 possono essere decrittografati. L'impostazione di questo parametro su consente `'V2_AND_LEGACY'` inoltre di decrittografare gli oggetti crittografati in un formato compatibile con V1. Per supportare la migrazione, impostare su. `@SecurityProfile` `'V2_AND_LEGACY'` Utilizzare `'V2'` solo per lo sviluppo di nuove applicazioni.

3. (opzionale) Includi il `@KmsAllowDecryptWithAnyCmk` parametro in `S3EncryptionClientV2::getObject` ed è stato aggiunto `S3EncryptionClientV2::getObjectAsync* methods.` un nuovo parametro chiamato`@KmsAllowDecryptWithAnyCmk`. L'impostazione di questo parametro `true` consente la decrittografia senza fornire una chiave KMS. Il valore predefinito è `false`.

4. Per la decrittografia con un client V2, se il `@KmsAllowDecryptWithAnyCmk` parametro non è impostato su `true` per le chiamate al `"getObject..."` metodo, è necessario fornire un `kms-key-id` al costruttore. `KmsMaterialsProviderV2`

Esempi di migrazione

Esempio 1: migrazione ai client V2

Premigrazione

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Post-migrazione

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);
```

Esempio 2: utilizzo con AWS KMS kms-key-id

> **Note**
>
> Questi esempi utilizzano importazioni e variabili definite nell'Esempio 1. Ad esempio, `$encryptionClient`.

Premigrazione

```
use Aws\Crypto\KmsMaterialsProvider;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProvider(
    new KmsClient([
        'profile' => 'default',
```

```
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

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

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

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

Post-migrazione

```
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,
]);
```

Domande frequenti sulla AWS SDK per PHP versione 3

Quali metodi sono disponibili su un client?

AWS SDK per PHP Utilizza descrizioni dei servizi e metodi dinamici magic __call () per eseguire operazioni API. Puoi trovare un elenco completo dei metodi disponibili per i client del servizio Web nella documentazione API del client.

Cosa devo fare in caso di errore cURL di un certificato SSL?

Questo problema può verificarsi quando si utilizza un pacchetto out-of-date CA con cURL e SSL. Puoi risolvere questo problema aggiornando il pacchetto CA sul tuo server o scaricando un altro pacchetto up-to-date CA direttamente dal sito web cURL.

Per impostazione predefinita, AWS SDK per PHP utilizzerà il pacchetto CA configurato al momento della compilazione di PHP. Puoi modificare il bundle CA predefinito utilizzato da PHP modificando l'impostazione di configurazione `openssl.cafile` del file PHP.ini impostandola sul percorso di un file CA sul disco.

Quali versioni dell'API sono disponibili per un client?

L'opzione `version` è obbligatoria durante la creazione di un client. <index.html>Un elenco delle versioni API disponibili è disponibile nella pagina di documentazione API di ciascun cliente:::aws-php-class. Se non riesci a caricare una versione dell'API specifica, potrebbe essere necessario aggiornare la copia dell' AWS SDK per PHP.

Puoi specificare la stringa `latest` nel valore di configurazione "versione" per utilizzare la versione dell'API più recente disponibile che il provider dell'API del client è in grado di trovare (l'api_provider predefinito analizzerà la directory `src/data` dell'SDK per cercare i modelli di API).

> **Warning**
>
> Non è consigliabile utilizzare `latest` in un'applicazione di produzione, perché l'importazione di una nuova versione secondaria dell'SDK che include un aggiornamento dell'API potrebbe interrompere il funzionamento di tale applicazione di produzione.

Quali versioni della regione sono disponibili per un client?

L'opzione `region` è obbligatoria durante la creazione di un client e viene specificata tramite un valore di stringa. Per un elenco delle AWS regioni e degli endpoint disponibili, consulta AWS Regioni ed endpoint in. Riferimenti generali di AWS

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

Perché non riesco a caricare o scaricare i file di dimensioni superiori a 2 GB?

Dal momento che il tipo di numero intero di PHP è firmato e molte delle piattaforme utilizzano i numeri interi a 32 bit, l' AWS SDK per PHP non è in grado di gestire correttamente i file di dimensioni superiori a 2 GB su uno stack a 32 bit (dove lo "stack" include CPU, sistema operativo, server Web e sistema binario PHP). Si tratta di un problema noto di PHP. Nel caso di Microsoft Windows, soltanto le build di PHP 7 supportano i numeri interi a 64 bit.

La soluzione consigliata è utilizzare uno stack di Linux a 64 bit, come l'AMI Amazon Linux a 64 bit, con la versione più recente di PHP installata.

Per ulteriori informazioni, consulta la sezione relativa alla dimensione dei file PHP e ai valori restituiti.

Come è possibile controllare i dati che vengono trasmessi in rete?

Utilizzando l'opzione `debug` in un costruttore di client, puoi ottenere le informazioni di debug, tra cui quelle relative ai dati trasmessi in rete. Se questa opzione è impostata su `true`, tutte le mutazioni del comando in esecuzione, la richiesta in fase di invio, la risposta in fase di ricezione e il risultato in fase di elaborazione vengono emessi in STDOUT. Sono inclusi i dati inviati e ricevuti in rete.

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

```
]);
```

Come è possibile impostare delle intestazioni arbitrarie su una richiesta?

Puoi aggiungere intestazioni arbitrarie a un'operazione di servizio aggiungendo un middleware personalizzato all'`Aws\HandlerList` di una `Aws\CommandInterface` o `Aws\ClientInterface`. L'esempio seguente mostra come aggiungere un'`X-Foo-Baz`intestazione a una specifica operazione di Amazon `PutObject` S3 utilizzando `Aws\Middleware::mapRequest` il metodo helper.

Consulta mapRequest per ulteriori informazioni.

Come posso firmare una richiesta arbitraria?

Puoi firmare una richiesta arbitraria:aws-php-class: PSR-7 <class-PSR.http.Message. RequestInterface.html> utilizzando la classe:: SignatureV4 dell'SDK. aws-php-class <class-Aws.Signature.SignatureV4.html>

Per un esempio completo di come eseguire questa operazione, consulta Firmare richieste di CloudSearch dominio Amazon personalizzate con la AWS SDK per PHP versione 3.

Come posso modificare un comando prima di inviarlo?

Puoi modificare un comando prima di inviarlo aggiungendo un middleware personalizzato all'`Aws \HandlerList` di una `Aws\CommandInterface` o `Aws\ClientInterface`. L'esempio seguente mostra come aggiungere dei parametri di comando personalizzati a un comando prima dell'invio, essenzialmente tramite l'aggiunta delle opzioni predefinite. In questo esempio viene utilizzato il metodo helper `Aws\Middleware::mapCommand`.

Consulta mapCommand per ulteriori informazioni.

Che cos'è un CredentialsException?

Se visualizzi un messaggio `Aws\Exception\CredentialsException` durante l'utilizzo di AWS SDK per PHP, significa che all'SDK non sono state fornite credenziali e che non è stato possibile trovare le credenziali nell'ambiente.

Se crei istanze di un client senza credenziali, la prima volta che effettui un'operazione di servizio, l'SDK tenterà di cercare le credenziali. Prima controlla alcune variabili di ambiente specifiche, quindi cerca le credenziali del profilo dell'istanza, che sono disponibili solo su EC2 istanze Amazon configurate. Se le credenziali non vengono specificate o trovate, viene generata una `Aws\Exception\CredentialsException`.

Se visualizzi questo errore e intendi utilizzare le credenziali del profilo dell'istanza, devi assicurarti che l' EC2 istanza Amazon su cui è in esecuzione l'SDK sia configurata con un ruolo IAM appropriato.

Se visualizzi questo errore e non intendi utilizzare le credenziali del profilo dell'istanza, devi assicurarti di specificare le credenziali corrette nell'SDK.

Per ulteriori informazioni, consulta Credenziali per la versione 3. AWS SDK per PHP

Funziona su HHVM? AWS SDK per PHP

Al momento AWS SDK per PHP non funziona su HHVM e non sarà in grado di farlo fino a quando il problema con la semantica del rendimento in HHVM non sarà risolto.

Come posso disabilitare il protocollo SSL?

Puoi disabilitare il protocollo SSL impostando il parametro `scheme` in un metodo client factory su "http". È importante tenere presente che non tutti i servizi supportano l'accesso `http`. Per un elenco AWS delle regioni, degli endpoint e degli Riferimenti generali di AWS schemi supportati, consulta Regioni ed endpoint nella sezione Regioni ed endpoint.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

> **Warning**
>
> Dal momento che, rispetto al protocollo TCP, il protocollo SSL richiede la crittografia di tutti i dati e più pacchetti TCP per il completamento di un handshake connessione, la sua disabilitazione può comportare un leggero miglioramento delle prestazioni. Tuttavia, con il protocollo SSL disabilitato, tutti i dati vengono trasmessi in rete senza essere crittografati.

Prima di disabilitare il protocollo SSL, valuta attentamente le implicazioni a livello di protezione e le probabilità di intercettazione sulla rete.

Cosa devo fare se visualizzo un "Errore di analisi"?

Il motore PHP genera degli errori di analisi quando incontra una sintassi che non è in grado di interpretare. Ciò si verifica quasi sempre quando tenti di eseguire il codice scritto per una versione diversa di PHP.

Se riscontri un errore di analisi, controlla il sistema e assicurati che soddisfi i requisiti e i consigli dell'SDK per la versione 3. AWS SDK per PHP

Perché il client Amazon S3 decomprime i file gzip?

Per impostazione predefinita, alcuni gestori HTTP, incluso il gestore predefinito Guzzle 6 HTTP, decomprimono i corpi della di risposta compressi. Puoi sostituire questo comportamento impostando l'opzione HTTP decode_content su `false`. Per motivi di compatibilità con le versioni precedenti, questa impostazione predefinita non può essere modificata, ma ti consigliamo di disabilitare la decodifica dei contenuti al livello del client S3.

Consulta decode_content per un esempio di come disabilitare la decodifica automatica dei contenuti.

Come faccio a disabilitare la firma del corpo in Amazon S3?

Puoi disabilitare la firma del corpo impostando il parametro `ContentSHA256` nell'oggetto comando su `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD`. Quindi lo AWS SDK per PHP utilizzerà come intestazione 'x-amz-content-sha-256' e come checksum del corpo nella richiesta canonica.

```
$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);
```

Come viene gestito lo schema dei nuovi tentativi nell' AWS SDK per PHP?

Ha un programma che gestisce il comportamento dei AWS SDK per PHP tentativi. `RetryMiddleware` In termini dei codici di stato HTTP 5xx relativi agli errori del server, l'SDK ritenta su 500, 502, 503 e 504.

Con i nuovi tentativi vengono gestite anche le eccezioni di throttling, incluse `RequestLimitExceeded`, `Throttling`, `ProvisionedThroughputExceededException`, `ThrottlingException`, `RequestThrottled` e `BandwidthLimitExceeded`.

AWS SDK per PHP Inoltre integra il ritardo esponenziale con un algoritmo di backoff e jitter nello schema di riprova. Inoltre, il comportamento di riprova predefinito è configurato come `3` per tutti i servizi ad eccezione di Amazon DynamoDB, che lo è. `10`

Come posso gestire le eccezioni con codici di errore?

Oltre alle `Exception` classi AWS SDK per PHP-customized, ogni client di AWS servizio ha una propria classe di eccezioni da cui eredita. AwsExceptionAwsException Puoi determinare i tipi di errore più specifici da individuare con gli errori specifici dell'API elencati nella sezione `Errors` di ciascun metodo.

Le informazioni sul codice di errore sono disponibili con 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;
    }
}
```

Glossario

Versione API

I servizi includono una o più versioni dell'API e la versione in uso determina quali operazioni e parametri sono validi. Le versioni dell'API presentano la formattazione di una data. Ad esempio, l'ultima versione dell'API per Amazon S3 è. `2006-03-01` Specificare una versione quando si configura un oggetto client.

Client

Gli oggetti client vengono utilizzati per eseguire operazioni per un servizio. Ogni servizio che è supportato in SDK ha un oggetto client corrispondente. Gli oggetti client hanno metodi che corrispondono one-to-one alle operazioni del servizio. Consulta la guida all'utilizzo di base per i dettagli su come creare e utilizzare gli oggetti client.

Comando

Gli oggetti di comando incapsulano l'esecuzione di un'operazione. Quando si seguono i modelli di utilizzo di base di SDK, non si gestiranno direttamente gli oggetti di comando. È possibile accedere agli oggetti di comando utilizzando il metodo `getCommand()` di un client, in modo da utilizzare caratteristiche avanzate di SDK, ad esempio le richieste simultanee e il batching. Per maggiori dettagli, consultate Command Objects nella guida alla AWS SDK per PHP versione 3.

Gestore

Un gestore è una funzione che esegue l'effettiva trasformazione di un comando e richiesta in un risultato. Un gestore in genere invia richieste HTTP. I gestori possono essere composti con middleware per potenziare il comportamento. Un gestore è una funzione che accetta un `Aws\CommandInterface` e un `Psr\Http\Message\RequestInterface` e restituisce una promessa che viene soddisfatta con un `Aws\ResultInterface` o respinta con un motivo `Aws\Exception\AwsException`.

JMESPath

JMESPathè un linguaggio di interrogazione per dati simili a JSON. AWS SDK per PHP Utilizza JMESPath espressioni per interrogare le strutture di dati PHP. JMESPath le espressioni possono essere utilizzate direttamente sugli `Aws\Result` `Aws\ResultPaginator` oggetti tramite il `search($expression)` metodo.

Middleware

Il middleware è uno speciale tipo di funzioni di alto livello che migliorano il comportamento di trasferimento di un comando e di delega a un gestore "successivo". Le funzioni middleware accettano `Aws\CommandInterface` e `Psr\Http\Message\RequestInterface` e restituiscono una promessa che viene soddisfatta con `Aws\ResultInterface` o respinta con un motivo `Aws\Exception\AwsException`.

Operazione

Si riferisce a una singola operazione all'interno dell'API di un servizio (ad esempio, `CreateTable` per DynamoDB, `RunInstances` per Amazon). EC2 In SDK, le operazioni vengono eseguite chiamando il metodo con lo stesso nome sull'oggetto client del servizio corrispondente. L'esecuzione di un'operazione prevede la preparazione e l'invio di una richiesta HTTP al servizio e l'analisi della risposta. Questo processo di esecuzione di un'operazione viene estrapolato da SDK tramite oggetti di comando.

Impaginatore

Alcune operazioni AWS di servizio sono suddivise in pagine e rispondono con risultati troncati. Ad esempio, l'`ListObjects`operazione di Amazon S3 restituisce solo fino a 1000 oggetti alla volta. Operazioni come queste richiedono l'esecuzione di richieste successive con parametri token (o marker) per recuperare l'intero set di risultati. Gli impaginatori sono una funzionalità dell'SDK che funge da astrazione rispetto a questo processo per semplificare l'utilizzo di paginated da parte degli sviluppatori. APIs Sono accessibili mediante il metodo `getPaginator()` del client. Per maggiori dettagli, consulta Paginators nella guida alla versione 3. AWS SDK per PHP

Promessa

Una promessa rappresenta l'eventuale risultato di un'operazione asincrona. La modalità principale di interazione con una promessa è attraverso il metodo "then", che registra callback per ricevere il valore eventuale di una promessa o il motivo per cui la promessa non può essere soddisfatta.

Regione

I servizi sono supportati in una o più regioni geografiche. I servizi possono avere endpoint diversi URLs in ogni regione, che servono a ridurre la latenza dei dati nelle applicazioni. Fornire una regione quando si configura un oggetto client, in modo che SDK possa determinare quale endpoint utilizzare con il servizio.

SDK

Il termine «SDK» può riferirsi alla AWS SDK per PHP libreria nel suo insieme, ma si riferisce anche alla **`Aws\Sdk`** classe (docs), che funge da fabbrica per gli oggetti client di ciascun servizio. La classe `Sdk` consente inoltre di fornire un set di valori di configurazione globali che vengono applicati a tutti gli oggetti client che crea.

Servizio

Un modo generico per fare riferimento a qualsiasi AWS servizio (ad esempio, Amazon S3, Amazon DynamoDB, ecc.) AWS OpsWorks. Ogni servizio presenta un oggetto client corrispondente in SDK che supporta una o più versioni dell'API. Ogni servizio include inoltre una o più operazioni che costituiscono l'API. I servizi sono supportati in una o più regioni.

Firma

Durante l'esecuzione di operazioni, SDK utilizza le tue credenziali per creare una firma digitale della tua richiesta. Quindi, il servizio verifica la firma prima di elaborare la richiesta. Il processo di firma viene incapsulato da SDK e viene eseguito automaticamente utilizzando le credenziali configurate per il client.

Waiter

Uno waiter è una caratteristica di SDK che semplifica l'utilizzo di operazioni che modificano lo stato di una risorsa ed è per sua natura coerente o asincrono. Ad esempio, l'operazione Amazon `CreateTable` DynamoDB invia immediatamente una risposta, ma la tabella potrebbe non essere pronta per l'accesso per diversi secondi. L'esecuzione di uno waiter consente di attendere fino a quando una risorsa non si trova in uno stato specifico mediante la sospensione e il polling dello stato della risorsa. È possibile accedere agli waiter utilizzando il metodo `waitUntil()` del client. Per maggiori dettagli, consulta la guida Waiters in the AWS SDK per PHP Version 3.

Per la AWS terminologia più recente, consulta il AWS Glossario in. Riferimenti generali di AWS

Cronologia dei documenti

Le tabelle seguenti descrivono le modifiche importanti dall'ultima versione della Developer Guide. AWS SDK per PHP

Le modifiche più recenti:

Modifica	Descrizione	Data
Revisioni degli argomenti relativi alle credenziali	Argomento riorganizzato. Sono stati forniti ulteriori dettagli per la catena di fornitori di credenziali predefinita.	10 gennaio 2025
Caricamenti multiparte di Amazon S3	Documenta l'array «params» che può essere utilizzato per configurare i sottocomandi di e `ObjectUploader` `MultipartUploader`	6 novembre 2024
Caricatore di oggetti	Chiarisci l'uso dei callback disponibili nei caricamenti per S3 `ObjectUploader`	11 ottobre 2024
Aggiorna i nomi dei bucket Amazon S3	Nomi dei bucket S3 aggiornati in tutta la guida.	30 settembre 2024
Endpoint EventBridge globali Amazon	Aggiungi un esempio di codice che mostra come usare gli endpoint EventBridge globali di Amazon	22 dicembre 2023
AWS Common Runtime (AWS CRT)	Aggiungi un argomento che discuta l'uso di AWS Common Runtime (AWS CRT) da parte dell'SDK for PHP.	17 novembre 2023

StreamWrapper mkdir () aggiorna	Aggiungi informazioni sull'util izzo di bucket e oggetti cartella utilizzando. `mkdir()`	2 novembre 2023
Creazione di client di servizio	Aggiorna i frammenti di codice rimuovendo il parametro «version» poiché l'imposta zione predefinita è «più recente».	31 agosto 2023
Sommario	Sommario aggiornato per rendere più accessibili gli esempi di codice.	1 giugno 2023
Aggiornamenti delle best practice di IAM	Guida aggiornata per l'allinea mento alle best practice IAM. Per ulteriori informazioni, consulta Best practice per la sicurezza in IAM. Aggiornam enti alla Guida introduttiva.	20 maggio 2023
Gestione trasferimenti Amazon S3	Aggiunta l'opzione `add_content_md5` di trasferimento.	13 aprile 2023
Caricamenti multiparte di Amazon S3	Informazioni di configura zione incluse per i caricamen ti sincroni. Aggiunta l'opzione di `add_content_md5` caricamento per i caricamenti asincroni.	13 aprile 2023

Informazioni di riferimento	Sono stati aggiunti diversi collegamenti ai contenuti di dettaglio pertinenti nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti . Formattazione della guida aggiornata.	14 settembre 2022
Pulizia generale	Sono stati aggiunti riferimenti alla AWS SDKs and Tools Reference Guide. AWS Key Management Service Sezioni aggiornate per riflettere gli aggiornamenti terminologici.	23 agosto 2022
Lavorare con i servizi AWS	Elenchi inclusi degli esempi di codice disponibili su GitHub.	1 aprile 2022
Abilitazione delle metriche SDK	Sono state rimosse le informazioni sull'abilitazione delle metriche SDK, che erano obsolete il 20 dicembre 2021.	27 gennaio 2022
Migrazione del client di crittografia Amazon S3	È stato aggiunto un argomento sulla migrazione del client di crittografia Amazon S3	7 agosto 2020

Modifiche precedenti:

Modifica	Descrizione	Data di rilascio
Esempi di Secrets Manager	Aggiunta di altri esempi di servizi	27 marzo 2019
Rilevamento di endpoint	Configurazione del rilevamento di endpoint	15 febbraio 2019

Modifica	Descrizione	Data di rilascio
Amazon CloudFront	Aggiunta di altri esempi di servizi	25 gennaio 2019
Caratteristiche del servizio	Parametri SDK	11 gennaio 2018
Amazon Kinesis, Amazon SNS	Aggiunta di altri esempi di servizi	14 dicembre 2018
Esempi di Amazon SES	Aggiunta di altri esempi di servizi	5 ottobre 2018
AWS KMS esempi	Aggiunta di altri esempi di servizi	8 agosto 2018
Credenziali	Chiarimento e semplificazione della guida delle credenziali	30 giugno 2018
MediaConvert esempi	Aggiunta di altri esempi di servizi	15 giugno 2018
Nuovo layout Web	Documentazione passata allo stile AWS	9 maggio 2018
Crittografia Amazon S3	Crittografia lato client	17 novembre 2017
Amazon S3, Amazon SQS	Aggiunta di altri esempi di servizi	26 marzo 2017
Amazon S3, IAM, Amazon EC2	Aggiunta di altri esempi di servizi	17 marzo 2017
Aggiunta di credenziali	Aggiunge il supporto per AssumeRole e ini	17 gennaio 2017
Esempi di S3	Post prefirmati e basati su più regioni di S3	18 marzo 2016

Modifica	Descrizione	Data di rilascio
OpenSearch Service e Amazon CloudSearch	Aggiunta di altri esempi di servizi	28 dicembre 2015
Riga di comando	Aggiunta di parametri di comando	13 agosto 2015
Caratteristiche del servizio	Aggiunge funzionalità di servizio per S3 e AWS	30 aprile 2015
Nuova versione dell'SDK	La versione 3 di quella AWS SDK per PHP rilasciata.	26 maggio 2015

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.