

Entwicklerhandbuch für Version 3

AWS SDK für PHP



AWS SDK für PHP: Entwicklerhandbuch für Version 3

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist das AWS SDK für PHP?	1
Beginnen Sie mit dem SDK	1
Weitere Ressourcen	1
API-Dokumentation	2
Wartung und Support für SDK-Hauptversionen	2
Erste Schritte	3
Voraussetzungen	3
Voraussetzungen	3
Empfehlungen	3
Kompatibilitätstest	5
Installation	5
Als Abhängigkeit über AWS SDK für PHP Composer installieren	5
Installation mithilfe des Paketes Phar	6
Installation mithilfe der ZIP-Datei	7
Authentifizierung mit AWS	7
Richten Sie die Authentifizierung mit IAM Identity Center für die lokale Entwicklung ein	8
Starten Sie eine AWS Access-Portal-Sitzung	9
Erfahren Sie mehr über Authentifizierung	10
Eine einfache Anwendung erstellen	10
Voraussetzungen	10
Das SDK in Ihren Code einbeziehen	11
Schreiben des Codes	11
Das Programm wird ausgeführt	12
Nächste Schritte	12
AWS Cloud9 Mit dem SDK verwenden	12
Schritt 1: Richten Sie Ihren AWS-Konto zur Verwendung ein AWS Cloud9	13
Schritt 2: Richten Sie Ihre AWS Cloud9 Entwicklungsumgebung ein	13
Schritt 3: Richten Sie das ein AWS SDK für PHP	14
Schritt 4: Laden Sie den Beispielcode herunter	15
Schritt 5: Beispielcode ausführen	16
Konfiguration von Service-Clients	18
Client-Konfiguration extern	18
Kette von Konfigurationsanbietern für die Client-Konfiguration	19
Erstellen Sie einen Service-Client, der mit externen Einstellungen konfiguriert wurde	20

AWS SDK für PHP Umgebungsvariablen der Version 3	20
Client-Konfiguration im Code	21
Grundkonfiguration im Code	21
Verwendung der Sdk Klasse	22
Konstruktor-Optionen	23
AWS-Region	53
Auflösungskette für Regionen	53
Bewährte Methoden	54
Anbieter von Anmeldeinformationen	55
Was ist ein Anmeldeinformationsanbieter in Version 3? AWS SDK für PHP	55
Grundlegendes zur Standard-Anbieterkette für Anmeldeinformationen in AWS SDK für PHP Version 3	56
Integrierte Anmeldeinformationsanbieter in der AWS SDK für PHP Version 3	57
Verkettung von Anmeldeinformationsanbietern im SDK for PHP	69
Erstellen eines benutzerdefinierten Anmeldeinformationsanbieters zur Verwendung mit dem SDK for PHP	70
Speichern von Anmeldeinformationen im SDK for PHP	71
Nehmen Sie eine IAM-Rolle an	72
Verwenden Sie temporäre Anmeldeinformationen aus AWS STS dem SDK for PHP	79
Anonyme Clients im SDK for PHP erstellen	82
Verwenden Sie die AWS CRT-Erweiterung	82
Benötige ich die AWS CRT-Erweiterung?	83
Wie installiere ich die AWS CRT-Erweiterung?	83
Verwenden der SDK	84
AWS-Service Anfragen stellen	84
Überblick über den Workflow für SDK-Anfragen	84
Einen Basis-Serviceclient erstellen	85
Senden von Anforderungen	86
Mit Ergebnisobjekten arbeiten	87
Befehlsobjekte	89
Asynchrone Programmierung	96
Promises	98
Fehlerbehandlung	105
Synchrone Fehlerbehandlung	105
Asynchrone Fehlerbehandlung	106
Handler und Middleware	107

Handler	108
Middleware	110
Benutzerdefinierte Handler erstellen	118
Streams	119
Stream-Dekorateur	119
Paginierung	124
Paginator-Objekte	124
Aufzählen von Daten aus Ergebnissen	125
Asynchrone Paginierung	126
Waiter	127
Konfiguration für Kellner	127
Asynchron warten	129
JMESPath Ausdrücke	130
Extrahieren von Daten aus Ergebnissen	131
Extrahieren von Daten aus Paginatoren	135
Ich rufe an AWS-Services	137
Nutzen Sie Funktionen und Optionen	137
Amazon DynamoDB	137
Amazon S3	145
Codebeispiele mit Anleitung	232
Anmeldeinformationen	232
CloudFront Amazon-Beispiele	233
Amazon CloudSearch	262
CloudWatch Amazon-Beispiele	265
EC2 Amazon-Beispiele	289
OpenSearch Amazon-Dienst	303
AWS Identity and Access Management Beispiele	304
AWS Key Management Service	329
Beispiele für Kinesis	352
AWS Elemental MediaConvert	369
Amazon S3 S3-Beispiele	376
AWS Secrets Manager	411
Amazon-SES-Beispiele	421
Beispiele für Amazon SNS	454
Beispiele für Amazon SQS	473
Amazon EventBridge	486

Codebeispiele	488
API Gateway	489
Aktionen	489
Szenarien	495
Aurora	495
Szenarien	495
Auto Scaling	496
Erste Schritte	497
Grundlagen	498
Aktionen	489
Amazon Bedrock	512
Aktionen	489
Amazon Bedrock Runtime	514
Szenarien	495
Amazon Nova	516
Amazon Titan Image Generator	518
Anthropic Claude	519
Stabile Diffusion	521
Amazon DocumentDB	522
Serverless-Beispiele	523
DynamoDB	524
Grundlagen	498
Aktionen	489
Szenarien	495
Serverless-Beispiele	523
Amazon EC2	557
Aktionen	489
AWS Glue	562
Grundlagen	498
Aktionen	489
IAM	582
Grundlagen	498
Aktionen	489
Kinesis	599
Serverless-Beispiele	523
AWS KMS	603

Erste Schritte	497
Grundlagen	498
Aktionen	489
Lambda	639
Grundlagen	498
Aktionen	489
Szenarien	495
Serverless-Beispiele	523
Amazon MSK	669
Serverless-Beispiele	523
Amazon RDS	671
Aktionen	489
Szenarien	495
Serverless-Beispiele	523
Amazon RDS Data Service	680
Szenarien	495
Amazon Rekognition	681
Szenarien	495
Amazon S3	682
Erste Schritte	497
Grundlagen	498
Aktionen	489
Szenarien	495
Serverless-Beispiele	523
S3-Verzeichnis-Buckets	705
Grundlagen	498
Amazon SES	721
Szenarien	495
Amazon SNS	722
Aktionen	489
Szenarien	495
Serverless-Beispiele	523
Amazon SQS	743
Serverless-Beispiele	523
Migration von Version 2	747
Einführung	747

Was ist neu in Version 3?	747
Entkoppelte HTTP-Schicht	747
Asynchrone Anfragen	96
Was ist anders als Version 2?	748
Die Projektabhängigkeiten werden aktualisiert	748
Regions- und Versionsoptionen sind jetzt erforderlich	749
Die Client-Instanziierung verwendet den Konstruktor	749
Die Client-Konfiguration hat sich geändert	750
Einige API-Ergebnisse haben sich geändert	751
Enum-Klassen wurden entfernt	754
Feinkörnige Ausnahmeklassen wurden entfernt	754
Statische Fassadenklassen wurden entfernt	755
Paginator ersetzen Iteratoren	755
Viele Abstraktionen auf höherer Ebene haben sich geändert	756
Vergleich von Codebeispielen aus beiden Versionen des SDK	757
Beispiel: Amazon S3 ListObjects S3-Betrieb	757
Beispiel: Instanzieren eines Clients mit globaler Konfiguration	759
Sicherheit	761
Datenschutz	761
Identitäts- und Zugriffsverwaltung	763
Zielgruppe	763
Authentifizierung mit Identitäten	764
Verwalten des Zugriffs mit Richtlinien	765
Wie AWS-Services arbeiten Sie mit IAM	767
Fehlerbehebung bei AWS Identität und Zugriff	767
Compliance-Validierung	770
Ausfallsicherheit	770
Infrastruktursicherheit	771
Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2)	772
Überblick über die Migration	772
Aktualisieren Sie bestehende Clients, um neue Formate lesen zu können	772
Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients auf V2	773
Beispiele für Migrationen	774
Migration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3)	777
Überblick über die Migration	777
V3-Konzepte verstehen	778

Aktualisieren Sie vorhandene Clients, damit sie neue Formate lesen können	780
Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf Version 3	782
Weitere Beispiele	786
Häufig gestellte Fragen	791
Welche Methoden sind auf einem Client verfügbar?	791
Was mache ich bei einem cURL SSL-Zertifikatsfehler?	791
Welche API-Versionen sind für einen Client verfügbar?	791
Welche Regionsversionen sind für einen Client verfügbar?	792
Warum kann ich keine Dateien mit Größen über 2 GB hoch- oder herunterladen?	792
Wie kann ich sehen, welche Daten übertragen wurden?	792
Wie kann ich zufällige Header für eine Anforderung festlegen?	793
Wie kann ich eine beliebige Anforderung signieren?	793
Wie kann ich einen Befehl vor dem Senden ändern?	793
Was ist ein CredentialsException?	793
AWS SDK für PHP Funktioniert das auf HHVM?	794
Wie deaktiviere ich SSL?	794
Was tue ich bei einem „Parse-Fehler“?	795
Warum dekomprimiert der Amazon S3 S3-Client gezippte Dateien?	795
Wie deaktiviere ich Body-Signing in Amazon S3?	795
Wie wird das Wiederholungsschema in AWS SDK für PHP behandelt?	796
Wie verarbeite ich Ausnahmen mit Fehlercodes?	796
Glossar	798
Dokumentverlauf	801
.....	dcccvii

Was ist die AWS SDK für PHP Version 3?

Die AWS SDK für PHP Version 3 ermöglicht es PHP-Entwicklern, [Amazon Web Services](#) in ihrem PHP-Code zu verwenden und mithilfe von Diensten wie Amazon S3, Amazon DynamoDB und Amazon Glacier robuste Anwendungen und Software zu entwickeln. Sie können innerhalb weniger Minuten loslegen, indem Sie das SDK über Composer installieren — indem Sie das `aws/aws-sdk-php` Paket benötigen — oder indem Sie die eigenständige Version [aws.zip](#) oder [aws.phar](#) Datei herunterladen.

Nicht alle Services sind sofort im SDK verfügbar. Informationen darüber, welche Dienste derzeit von der unterstützt werden AWS SDK für PHP, finden Sie unter [Dienstname und API-Version](#).

Note

Wenn Sie Ihren Code von Version 2 des SDK auf Version 3 migrieren, lesen Sie unbedingt den Artikel [Upgrade von Version 2 von](#). AWS SDK für PHP

Beginnen Sie mit dem SDK

Wenn Sie bereit sind, das SDK in der Praxis auszuprobieren, folgen Sie dem [Erste Schritte mit der AWS SDK für PHP Version 3](#) Kapitel. Es führt Sie durch die Authentifizierung mit Amazon S3 AWS, die Einrichtung Ihrer Entwicklungsumgebung und die Erstellung Ihrer ersten Basisanwendung.

Weitere Ressourcen

- [HÄUFIG GESTELLTE FRAGEN](#)
- [Glossar](#)
- [AWS SDKs Referenzhandbuch für Tools und Tools](#): Enthält Einstellungen, Funktionen und andere grundlegende Konzepte, die allen gemeinsam sind. AWS SDKs
- [Guzzle Documentation](#)
- Codebeispiele, die den AWS SDK für PHP verwenden, sind im [aws-doc-sdk-examplesawsdocs/-Repo](#) verfügbar.
- [PHP SDK-Community](#) auf Gitter.
- [AWS re:Post](#).

GitHub:

- Der Quellcode für AWS SDK für PHP ist im [aws-sdk-phpaws/Repo](#) verfügbar.
- [Beitrag zum SDK](#)
- [Melden von Fehlern oder Anfordern von Leistungsmerkmalen](#)

API-Dokumentation

Die API-Dokumentation für das SDK finden Sie unter [latest/reference/](#). <https://docs.aws.amazon.com/sdk-for-php/>

Wartung und Support für SDK-Hauptversionen

[Informationen zur Wartung und zum Support für SDK-Hauptversionen und die ihnen zugrunde liegenden Abhängigkeiten finden Sie im Referenzhandbuch und im Tools-Referenzhandbuch:AWS SDKs](#)

- [AWS SDKs und Richtlinien zur Wartung von Tools](#)
- [AWS SDKs und Matrix zur Unterstützung der Tools-Version](#)

Erste Schritte mit der AWS SDK für PHP Version 3

In diesem Kapitel erfahren Sie, wie Sie mit der AWS SDK für PHP Version 3 loslegen können.

Themen

- [Anforderungen und Empfehlungen für die AWS SDK für PHP Version 3](#)
- [Installation der AWS SDK für PHP Version 3](#)
- [Authentifizierung AWS mit AWS SDK für PHP Version 3](#)
- [Erstellen einer einfachen Anwendung mit der AWS SDK für PHP Version 3](#)
- [Zur Verwendung AWS Cloud9 mit der AWS SDK für PHP Version 3](#)

Anforderungen und Empfehlungen für die AWS SDK für PHP Version 3

Um optimale Ergebnisse zu erzielen AWS SDK für PHP, stellen Sie sicher, dass Ihre Umgebung die folgenden Anforderungen und Empfehlungen unterstützt.

Voraussetzungen

Um das verwenden zu können AWS SDK für PHP, müssen Sie PHP Version 8.1 oder höher mit aktivierter [SimpleXML PHP-Erweiterung](#) verwenden. Wenn Sie private Amazon signieren müssen CloudFront URLs, benötigen Sie auch die [OpenSSL-PHP-Erweiterung](#).

Empfehlungen

Zusätzlich zu den Mindestanforderungen empfehlen wir Ihnen, Folgendes zu installieren, zu deinstallieren und zu verwenden.

Installieren Sie [cURL](#) 7.16.2 oder höher.

Verwenden Sie eine aktuelle Version von cURL, die mit OpenSSL/NSS und zlib kompiliert wurde. Wenn cURL nicht auf Ihrem System installiert ist und Sie keinen benutzerdefinierten http_handler für Ihren Client konfigurieren, verwendet das SDK den PHP-Stream-Wrapper.

Verwenden von [OPCache](#)

Verwenden Sie die OPcache Erweiterung, um die PHP-Leistung zu verbessern, indem Sie vorkompilierten Skript-Bytecode im gemeinsamen Speicher speichern. Dadurch muss PHP keine Skripts mehr für jede Anforderung laden und analysieren. Diese Erweiterung ist standardmäßig aktiviert.

Wenn Sie Amazon Linux ausführen, müssen Sie das php56-opcache- oder php55-opcache yum-Paket installieren, um die Erweiterung verwenden zu können. OPCache

[Deinstallieren](#) Sie Xdebug in Produktionsumgebungen

Xdebug kann helfen, Leistungsengpässe zu identifizieren. Wenn die Leistung für Ihre Anwendung jedoch von entscheidender Bedeutung ist, installieren Sie die Xdebug-Erweiterung in Ihrer Produktionsumgebung nicht. Das Laden der Erweiterung verlangsamt die SDK-Leistung erheblich.

Verwenden Sie einen [Composer](#) Classmap Autoloader

Autoloader laden Klassen, wie sie von einem PHP-Skript benötigt werden. Composer generiert einen Autoloader, der die PHP-Skripts Ihrer Anwendung und alle anderen von Ihrer Anwendung benötigten PHP-Skripte automatisch laden kann, einschließlich der AWS SDK für PHP.

Für Produktionsumgebungen empfehlen wir die Verwendung eines Classmap-Autoloaders, um die Autoloader-Leistung zu verbessern. Sie können einen Classmap-Autoloader generieren, indem Sie die Option `-o` oder `--optimize-autoloader` an den Installationsbefehl von Composer übergeben.

Kompatibilitätstest

Führen Sie die [compatibility-test.php](#) Datei in der SDK-Codebasis aus, um zu überprüfen, ob Ihr System das SDK ausführen kann. Zusätzlich zur Erfüllung der SDK-Mindestanforderungen prüft der Kompatibilitätstest optionale Einstellungen und gibt Empfehlungen zur Verbesserung der Leistung. Der Kompatibilitätstest gibt die Ergebnisse entweder an die Befehlszeile oder einen Webbrowser aus. Beim Überprüfen der Testergebnisse in einem Browser werden erfolgreiche Überprüfungen grün, Warnungen lila und Fehler rot angezeigt. Wenn es über die Befehlszeile ausgeführt wird, wird das Ergebnis einer Prüfung in einer separaten Zeile angezeigt.

Wenn Sie ein Problem mit dem SDK melden, hilft die Freigabe der Ausgabe des Kompatibilitätstests, die zugrunde liegende Ursache zu identifizieren.

Installation der AWS SDK für PHP Version 3

Sie können die AWS SDK für PHP Version 3 installieren:

- Als Abhängigkeit per Composer
- Als vorkonfiguriertes phar des SDK
- Als ZIP-Datei des SDK

Bevor Sie AWS SDK für PHP Version 3 installieren, stellen Sie sicher, dass Ihre Umgebung PHP Version 8.1 oder höher verwendet. Erfahren Sie mehr über [Umgebungsanforderungen und Empfehlungen](#).

Note

Für die Installation des SDK über die Methoden .phar und .zip muss die [Multibyte String PHP-Erweiterung separat](#) installiert und aktiviert werden.

Als Abhängigkeit über AWS SDK für PHP Composer installieren

Composer ist die empfohlene Methode zur Installation von AWS SDK für PHP. Composer ist ein Tool für PHP, das die Abhängigkeiten Ihres Projekts verwaltet und installiert.

Weitere Informationen zur Installation von Composer, zur Konfiguration von Autoloading und zu anderen bewährten Verfahren zur Definition von Abhängigkeiten finden Sie unter [getcomposer.org](#).

Installieren von Composer

Wenn Composer noch nicht in Ihrem Projekt enthalten ist, laden Sie Composer auf der [Seite Composer herunterladen herunter und installieren Sie ihn](#).

- Folgen Sie für Windows den Anweisungen von Windows Installer.
- Folgen Sie für Linux den Installationsanweisungen in der Befehlszeile.

Über AWS SDK für PHP Composer als Abhängigkeit hinzufügen

Wenn [Composer bereits global auf Ihrem System installiert ist](#), führen Sie Folgendes im Basisverzeichnis Ihres Projekts aus, um es AWS SDK für PHP als Abhängigkeit zu installieren:

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

Geben Sie andernfalls diesen Composer-Befehl ein, um die neueste Version von AWS SDK für PHP als Abhängigkeit zu installieren.

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

Autoloader Ihren php-Skripts hinzufügen

Durch die Installation von Composer werden mehrere Ordner und Dateien in Ihrer Umgebung erstellt. Die primäre Datei, die Sie verwenden, ist `autoload.php`. Sie befindet sich im `vendor`-Ordner in Ihrer Umgebung.

Um das AWS SDK für PHP in Ihren Skripten zu verwenden, fügen Sie den Autoloader wie folgt in Ihre Skripts ein.

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

Installation mithilfe des Paketes Phar

Jede Version von AWS SDK für PHP enthält ein vorkonfiguriertes Phar-Archiv (PHP-Archiv), das alle Klassen und Abhängigkeiten enthält, die Sie für die Ausführung des SDK benötigen. Darüber hinaus registriert der Phar automatisch einen Klassen-Autoloader für die AWS SDK für PHP und all ihre Abhängigkeiten.

Sie können [das vorkonfigurierte phar herunterladen](#) und in Ihre Skripts einbinden.

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

Note

Die Verwendung von PHP mit dem Suhosin-Patch wird nicht empfohlen, ist aber auf Ubuntu- und Debian-Distributionen üblich. In diesem Fall müssen Sie möglicherweise die Verwendung von phars in der suhosin.ini aktivieren. Wenn Sie dies nicht tun, wird die Einbindung einer phar-Datei in Ihren Code einen stillen Fehler verursachen. Um suhosin.ini zu ändern, fügen Sie die folgende Zeile hinzu.

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

Installation mithilfe der ZIP-Datei

Das AWS SDK für PHP beinhaltet eine ZIP-Datei mit allen Klassen und Abhängigkeiten, die Sie zum Ausführen des SDK benötigen. Zusätzlich enthält die ZIP-Datei einen Klassen-Autoloader für AWS SDK für PHP und seine Abhängigkeiten.

Für die Installation des SDK [laden Sie die ZIP-Datei herunter](#) und extrahieren sie in Ihrem Projekt an einem von Ihnen angegebenen Speicherort. Fügen Sie dann den Autoloader wie folgt in Ihre Skripts ein.

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

Authentifizierung AWS mit AWS SDK für PHP Version 3

Sie müssen AWS bei der Entwicklung mit AWS-Services festlegen, wie sich Ihr Code authentifiziert. Sie können den programmatischen Zugriff auf AWS Ressourcen je nach Umgebung und verfügbarem AWS Zugriff auf unterschiedliche Weise konfigurieren.

Informationen zur Auswahl Ihrer Authentifizierungsmethode und deren Konfiguration für das SDK finden Sie unter [Authentifizierung und Zugriff](#) im AWS SDKs Referenzhandbuch zu Tools.

Richten Sie die Authentifizierung mit IAM Identity Center für die lokale Entwicklung ein

Wir empfehlen, dass neue Benutzer, die sich lokal entwickeln und von ihrem Arbeitgeber keine Authentifizierungsmethode erhalten, die Einrichtung einrichten AWS IAM Identity Center. Diese Methode beinhaltet die Installation von, AWS CLI um die Konfiguration zu vereinfachen und sich regelmäßig beim AWS Zugangsportal anzumelden. Wenn Sie sich für diese Methode entscheiden, sollte Ihre Umgebung die folgenden Elemente enthalten, nachdem Sie das Verfahren für die [IAM Identity Center-Authentifizierung](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch abgeschlossen haben:

- Die AWS CLI, mit der Sie eine AWS Access-Portal-Sitzung starten, bevor Sie Ihre Anwendung ausführen.
- Eine [gemeinsam genutzte AWSconfig Datei](#) mit einem [default] Profil mit einer Reihe von Konfigurationswerten, auf die das SDK verweisen kann. Informationen zum Speicherort dieser Datei finden Sie unter [Speicherort der gemeinsam genutzten Dateien](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
- Die gemeinsam genutzte config Datei enthält die [region](#)Einstellung. Dies legt die Standardeinstellung AWS-Region fest, die das SDK für Anfragen verwendet. Diese Region wird für SDK-Dienstanforderungen verwendet, die nicht explizit mit einer region Eigenschaft konfiguriert sind.
- Das SDK verwendet die [Konfiguration des SSO-Token-Anbieters](#) des Profils, um Anmeldeinformationen abzurufen, bevor Anfragen an gesendet AWS werden. Der sso_role_name Wert, bei dem es sich um eine IAM-Rolle handelt, die mit einem IAM Identity Center-Berechtigungssatz verbunden ist, ermöglicht den Zugriff auf die in Ihrer AWS-Services Anwendung verwendeten.

Die folgende config Beispieldatei zeigt ein Standardprofil, das mit der Konfiguration des SSO-Token-Anbieters eingerichtet wurde. Die sso_session Einstellung des Profils bezieht sich auf den benannten [sso-sessionAbschnitt](#). Der sso-session Abschnitt enthält Einstellungen zum Initiieren einer AWS Access-Portal-Sitzung.

```
[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
```

Für die Verwendung der IAM Identity Center-Authentifizierung müssen Ihrer Anwendung AWS SDK für PHP keine zusätzlichen Pakete (wie SSO undSSOIDC) hinzugefügt werden.

Starten Sie eine AWS Access-Portal-Sitzung

Bevor Sie eine Zugriffsanwendung ausführen AWS-Services, benötigen Sie eine aktive AWS Access-Portal-Sitzung, damit das SDK die IAM Identity Center-Authentifizierung zur Auflösung von Anmeldeinformationen verwenden kann. Abhängig von Ihrer konfigurierten Sitzungsdauer läuft Ihr Zugriff irgendwann ab und das SDK wird auf einen Authentifizierungsfehler stoßen. Um sich beim AWS Zugriffsportal anzumelden, führen Sie den folgenden Befehl in der aus AWS CLI.

```
aws sso login
```

Wenn Sie die Anweisungen befolgt haben und ein Standardprofil eingerichtet haben, müssen Sie den Befehl nicht mit einer `--profile` Option aufrufen. Wenn die Konfiguration Ihres SSO-Token-Anbieters ein benanntes Profil verwendet, lautet der Befehl `aws sso login --profile named-profile`.

Führen Sie den folgenden AWS CLI Befehl aus, um optional zu testen, ob Sie bereits eine aktive Sitzung haben.

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

Wenn Ihre Sitzung aktiv ist, werden in der Antwort auf diesen Befehl das in der gemeinsam genutzten config Datei konfigurierte IAM Identity Center-Konto und der Berechtigungssatz gemeldet.

 Note

Wenn Sie bereits eine aktive AWS Access-Portal-Sitzung haben und ausführen `aws sso login`, müssen Sie keine Anmeldeinformationen angeben.

Beim Anmeldevorgang werden Sie möglicherweise aufgefordert, den AWS CLI Zugriff auf Ihre Daten zu gewähren. Da AWS CLI das auf dem SDK für Python aufbaut, können Berechtigungsnachrichten Variationen des `botocore` Namens enthalten.

Erfahren Sie mehr über Authentifizierung

- Weitere Informationen zur Verwendung von IAM Identity Center für die Authentifizierung finden Sie unter [Grundlegendes zur IAM Identity Center-Authentifizierung](#) im AWS SDKs Referenzhandbuch zu Tools
- Weitere Informationen zu bewährten Methoden finden Sie unter [Bewährte Methoden für die Sicherheit in IAM](#) im IAM-Benutzerhandbuch.
- Informationen zur Erstellung kurzfristiger AWS Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldedaten](#) im IAM-Benutzerhandbuch.
- Weitere Informationen zu anderen Anbietern von Anmeldeinformationen, die Sie verwenden AWS SDK für PHP können, finden Sie unter [Standardisierte Anmeldeinformationsanbieter](#) im Referenzhandbuch AWS SDKs zu Tools.

Erstellen einer einfachen Anwendung mit der AWS SDK für PHP Version 3

Begrüßen Sie Amazon S3 mit dem AWS SDK für PHP. Im folgenden Beispiel wird eine Liste Ihrer Amazon S3 S3-Buckets angezeigt.

Voraussetzungen

- [Laden Sie das SDK herunter und installieren Sie es](#)
- Bevor Sie das verwenden können AWS SDK für PHP, müssen Sie die Authentifizierung mit einrichten AWS. Informationen zum Einrichten der Authentifizierung finden Sie unter [Authentifizierung AWS mit AWS SDK für PHP Version 3](#)

Das SDK in Ihren Code einbeziehen

Unabhängig davon, mit welcher Technik Sie das SDK installiert haben, können Sie das SDK mit nur einer `require`-Anweisung in den Code einfügen. In der folgenden Tabelle finden Sie den PHP-Code, der am besten zu Ihrer Installationstechnik passt. Ersetzen Sie alle Instances von `/path/to/` durch den tatsächlichen Pfad auf Ihrem System.

Installationstechnik	Anweisung anfordern
Verwenden von Composer	<code>require '/path/to/vendor/autoload.php';</code>
Verwenden von phar	<code>require '/path/to/aws.phar';</code>
Verwenden der ZIP	<code>require '/path/to/aws-auto-loader.php';</code>

In diesem Thema gehen wir von der Composer-Installationsmethode aus. Wenn Sie eine andere Installationsmethode verwenden, können Sie in diesem Abschnitt nach dem richtigen `require`-Code suchen.

Schreiben des Codes

Stellen Sie sicher, dass Sie sich authentifizieren können.

Kopieren Sie den folgenden Code und fügen Sie ihn in eine neue Quelldatei ein. Speichern und benennen Sie die Datei `hello-s3.php`.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;

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

//Create a S3Client
// snippet-start:[s3.php.list_buckets.main]
```

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

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

Das Programm wird ausgeführt

Öffnen Sie eine Eingabeaufforderung, um Ihr PHP-Programm auszuführen. Die typische Befehlssyntax zum Ausführen eines PHP-Programms lautet:

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

Dieser Beispielcode verwendet keine Argumente. Um diesen Code auszuführen, geben Sie Folgendes in die Befehlszeile ein:

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

Nächste Schritte

Um viele andere Amazon S3 S3-Operationen zu testen, schauen Sie sich das [AWS Code Examples Repository](#) auf an GitHub.

Zur Verwendung AWS Cloud9 mit der AWS SDK für PHP Version 3

Note

AWS Cloud9 ist für Neukunden nicht mehr verfügbar. Bestandskunden von AWS Cloud9 können den Service weiterhin wie gewohnt nutzen. [Weitere Informationen](#).

AWS Cloud9 ist eine webbasierte integrierte Entwicklungsumgebung (IDE), die eine Sammlung von Tools enthält, mit denen Sie Software in der Cloud programmieren, erstellen, ausführen, testen,

debuggen und veröffentlichen können. Sie können sie verwenden AWS Cloud9 AWS SDK für PHP , um Ihren PHP-Code mithilfe eines Browsers zu schreiben und auszuführen. AWS Cloud9 beinhaltet Tools wie einen Code-Editor und ein Terminal. Da die AWS Cloud9 IDE cloudbasiert ist, können Sie von Ihrem Büro, zu Hause oder von überall aus an Ihren Projekten arbeiten, indem Sie einen mit dem Internet verbundenen Computer verwenden. Allgemeine Informationen zu AWS Cloud9 finden Sie im [AWS Cloud9 Benutzerhandbuch](#).

Folgen Sie diesen Anweisungen zur Einrichtung AWS Cloud9 mit AWS SDK für PHP:

- [Schritt 1: Richten Sie Ihre AWS-Konto zur Verwendung ein AWS Cloud9](#)
- [Schritt 2: Richten Sie Ihre AWS Cloud9 Entwicklungsumgebung ein](#)
- [Schritt 3: Richten Sie AWS SDK für PHP](#)
- [Schritt 4: Laden Sie den Beispielcode herunter](#)
- [Schritt 5: Beispielcode ausführen](#)

Schritt 1: Richten Sie Ihren AWS-Konto zur Verwendung ein AWS Cloud9

Um zu verwenden AWS Cloud9, melden Sie sich AWS Cloud9 über die Konsole an AWS-Managementkonsole.

Note

Wenn Sie die Authentifizierung verwenden AWS IAM Identity Center , müssen Sie möglicherweise die erforderliche Berechtigung für zur vom Benutzer `iam:ListInstanceProfilesForRole` verknüpften Richtlinie in der IAM-Konsole hinzufügen.

Informationen zum Einrichten einer IAM-Entität in Ihrem AWS Konto für den Zugriff auf AWS Cloud9 und die Anmeldung an der AWS Cloud9 Konsole finden Sie unter [Team-Setup für AWS Cloud9 im Benutzerhandbuch](#).AWS Cloud9

Schritt 2: Richten Sie Ihre AWS Cloud9 Entwicklungsumgebung ein

Nachdem Sie sich bei der AWS Cloud9 Konsole angemeldet haben, verwenden Sie die Konsole, um eine AWS Cloud9 Entwicklungsumgebung zu erstellen. Nachdem Sie die Umgebung erstellt haben, AWS Cloud9 wird die IDE für diese Umgebung geöffnet.

Einzelheiten finden Sie unter [Erstellen einer Umgebung AWS Cloud9 im AWS Cloud9 Benutzerhandbuch](#).

Note

Wenn Sie Ihre Umgebung zum ersten Mal in der Konsole erstellen, empfehlen wir Ihnen, die Option Neue Instanz für environment (EC2) erstellen zu wählen. Diese Option weist an, eine Umgebung AWS Cloud9 zu erstellen, eine EC2 Amazon-Instance zu starten und dann die neue Instance mit der neuen Umgebung zu verbinden. Dies ist der schnellste Weg, um mit der Nutzung zu beginnen AWS Cloud9.

Wenn das Terminal in der IDE noch nicht geöffnet ist, öffnen Sie es. Wählen Sie auf der Menüleiste in der IDE Window, New Terminal (Fenster, Neues Terminal). Sie können das Terminalfenster verwenden, um Tools zu installieren und Ihre Anwendungen zu erstellen.

Schritt 3: Richten Sie das ein AWS SDK für PHP

Nachdem Sie die IDE für Ihre Entwicklungsumgebung AWS Cloud9 geöffnet haben, verwenden Sie das Terminalfenster, um die AWS SDK für PHP in Ihrer Umgebung einzurichten.

Composer ist die empfohlene Methode zur Installation von AWS SDK für PHP. Composer ist ein Tool für PHP, das die Abhängigkeiten Ihres Projekts verwaltet und installiert.

Weitere Informationen zur Installation von Composer, zur Konfiguration von Autoloading und zu anderen bewährten Verfahren zur Definition von Abhängigkeiten finden Sie unter getcomposer.org.

Installieren von Composer

Wenn Composer noch nicht in Ihrem Projekt enthalten ist, laden Sie Composer auf der [Seite Composer herunterladen herunter und installieren Sie ihn](#).

- Folgen Sie für Windows den Anweisungen von Windows Installer.
- Folgen Sie für Linux den Installationsanweisungen in der Befehlszeile.

Über AWS SDK für PHP Composer als Abhängigkeit hinzufügen

Wenn [Composer bereits global auf Ihrem System installiert ist](#), führen Sie Folgendes im Basisverzeichnis Ihres Projekts aus, um es AWS SDK für PHP als Abhängigkeit zu installieren:

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

Geben Sie andernfalls diesen Composer-Befehl ein, um die neueste Version von AWS SDK für PHP als Abhängigkeit zu installieren.

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

Autoloader Ihren php-Skripts hinzufügen

Durch die Installation von Composer werden mehrere Ordner und Dateien in Ihrer Umgebung erstellt. Die primäre Datei, die Sie verwenden, ist `autoload.php`. Sie befindet sich im `vendor`-Ordner in Ihrer Umgebung.

Um das AWS SDK für PHP in Ihren Skripten zu verwenden, fügen Sie den Autoloader wie folgt in Ihre Skripts ein.

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

Schritt 4: Laden Sie den Beispielcode herunter

Verwenden Sie das Terminalfenster, um den Beispielcode für AWS SDK für PHP in die AWS Cloud9 Entwicklungsumgebung herunterzuladen.

Führen Sie den folgenden Befehl aus, um eine Kopie aller in der offiziellen AWS SDK-Dokumentation verwendeten Codebeispiele in das Stammverzeichnis Ihrer Umgebung herunterzuladen:

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

Die Codebeispiele für AWS SDK für PHP befinden sich im `ENVIRONMENT_NAME/aws-doc-sdk-examples/php` Verzeichnis, in dem `ENVIRONMENT_NAME` sich der Name Ihrer Entwicklungsumgebung befindet.

Um den Vorgang anhand eines Amazon S3-Beispiels nachzuvollziehen, empfehlen wir, mit einem Codebeispiel zu beginnen `ENVIRONMENT_NAME/aws-doc-sdk-examples/php/example_code/s3/ListBuckets.php`. In diesem Beispiel werden Ihre Amazon S3 S3-Buckets aufgelistet.

Verwenden Sie das Terminalfenster, um zum s3 Verzeichnis zu navigieren und die Dateien aufzulisten.

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

Um die Datei zu öffnen AWS Cloud9, können Sie `ListBuckets.php` direkt im Terminalfenster auf das klicken.

Weitere Informationen zum Verständnis von Codebeispielen finden Sie unter [AWS SDK für PHP Codebeispiele](#).

Schritt 5: Beispielcode ausführen

Um Code in Ihrer AWS Cloud9 Entwicklungsumgebung auszuführen, wählen Sie in der oberen Menüleiste die Schaltfläche Ausführen. AWS Cloud9 erkennt automatisch die `.php` Dateierweiterung und verwendet den PHP-Runner (integrierter Webserver), um den Code auszuführen. Für dieses Beispiel wollen wir jedoch eigentlich die Option PHP (**cli**). Weitere Informationen zum Ausführen von Code in AWS Cloud9 finden Sie unter [Run Your Code](#) im AWS Cloud9 Benutzerhandbuch.

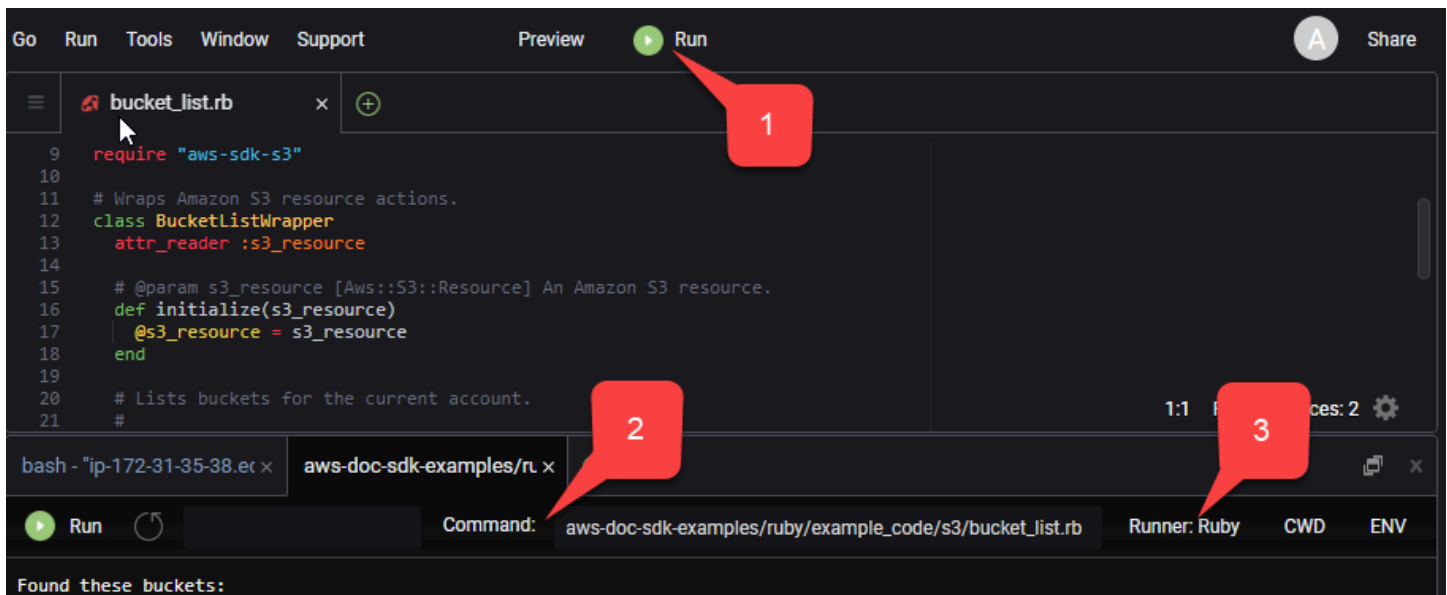
Beachten Sie im folgenden Screenshot die folgenden grundlegenden Bereiche:

- 1: Lauf. Die Schaltfläche Ausführen befindet sich in der oberen Menüleiste. Dadurch wird eine neue Registerkarte für Ihre Ergebnisse geöffnet.

Note

Sie können neue Laufkonfigurationen auch manuell erstellen. Wählen Sie auf der Menüleiste Run (Ausführen), Run Configurations (Run-Konfigurationen), New Run Configuration (Neue Run-Konfiguration) aus.

- 2: Befehl. AWS Cloud9 füllt das Befehlstextfeld mit dem Pfad und dem Dateinamen der Datei, die Sie ausführen. Wenn Ihr Code erwartet, dass Befehlszeilenparameter übergeben werden, können diese der Befehlszeile auf die gleiche Weise hinzugefügt werden, wie Sie es tun würden, wenn Sie den Code über ein Terminalfenster ausführen würden.
- 3: Läufer. AWS Cloud9 erkennt, dass Ihre Dateierweiterung lautet, `.php` und wählt den PHP-Runner (integrierter Webserver) aus, um Ihren Code auszuführen. Wählen Sie stattdessen PHP (**cli**), um dieses Beispiel auszuführen.



Jede Ausgabe, die aus dem laufenden Code generiert wurde, wird auf der Registerkarte angezeigt.

Konfiguration von Service-Clients in der AWS SDK für PHP Version 3

Für den AWS-Services programmgesteuerten Zugriff verwendet AWS SDK für PHP Version 3 jeweils ein Client-Objekt. AWS-Service Wenn Ihre Anwendung beispielsweise auf Amazon zugreifen muss EC2, erstellen Sie ein EC2 Amazon-Client-Objekt — eine Instance der Klasse [Ec2Client](#) — als Schnittstelle zu diesem Service. Anschließend verwenden Sie den Service-Client, um Anfragen an dieses zu stellen. AWS-Service

Es gibt viele Möglichkeiten, das SDK-Verhalten zu konfigurieren, aber letztendlich hat alles mit dem Verhalten von Service-Clients zu tun. Jede Konfiguration hat keine Wirkung, bis Ihr Code einen Service-Client erstellt, der die Konfiguration verwendet.

Beispiele für die Konfiguration, die Sie bereitstellen, sind:

- Wie sich Ihr Code authentifiziert AWS , wenn Sie einen Dienst aufrufen
- Den AWS-Region Sie möchten, dass ein Service-Client verwendet
- Einstellungen für Wiederholungsversuche und Timeout für Serviceanrufe
- HTTP-Proxykonfiguration

Einstellungen, Funktionen [AWS SDKs und andere grundlegende Konzepte, die vielen der Tools gemeinsam sind, finden Sie im Referenzhandbuch](#) und im Tools-Referenzhandbuch. AWS SDKs

Service-Clients für die AWS SDK für PHP Version 3 extern konfigurieren

Viele Konfigurationseinstellungen können außerhalb Ihres Codes bearbeitet werden. Die meisten Konfigurationseinstellungen können entweder als Umgebungsvariablen oder in einer separaten gemeinsam genutzten AWS config Datei festgelegt werden. Die AWS gemeinsam genutzte config Datei kann separate Einstellungssätze, sogenannte Profile, enthalten, um unterschiedliche Konfigurationen für verschiedene Umgebungen oder Tests bereitzustellen. Eine vollständige Beschreibung der gemeinsam AWS genutzten credentials Dateien config und Dateien finden Sie im [Referenzhandbuch zu Tools AWS SDKs und Tools](#).

Die meisten Umgebungsvariablen und Einstellungen für gemeinsam genutzte `config` Dateien sind standardisiert und werden von allen AWS SDKs Tools gemeinsam genutzt, um konsistente Funktionen in verschiedenen Programmiersprachen und Anwendungen zu gewährleisten.

Alle Einstellungen, die das SDK anhand der Umgebungsvariablen oder Konfigurationsdateien auflösen kann, finden Sie in der Referenz zu [Einstellungen im Referenzhandbuch AWS SDKs](#) und im Tools-Referenzhandbuch.

Kette von Konfigurationsanbietern für die Client-Konfiguration

Das SDK sucht an mehreren Stellen (oder Quellen) nach Konfigurationswerten.

1. Jede explizite Einstellung, die im Code oder auf einem Service-Client selbst festgelegt ist, hat Vorrang vor allen anderen Einstellungen.
2. Umgebungsvariablen
 - Einzelheiten zum Setzen von Umgebungsvariablen finden Sie unter [Umgebungsvariablen](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
 - Beachten Sie, dass Sie Umgebungsvariablen für eine Shell auf verschiedenen Gültigkeitsebenen konfigurieren können: systemweit, benutzerweit und für eine bestimmte Terminalsitzung.
3. Geteilte Dateien und Dateien `config credentials`
 - Einzelheiten zum Einrichten dieser Dateien finden Sie im Referenzhandbuch „[Gemeinsam genutzte configcredentials Dateien](#)“ AWS SDKs und „Tools“.
4. Jeder Standardwert, der vom SDK-Quellcode selbst bereitgestellt wird, wird zuletzt verwendet.
 - Für einige Eigenschaften, wie z. B. Region, gibt es keine Standardeinstellung. Sie müssen sie entweder explizit im Code, in einer Umgebungseinstellung oder in der gemeinsam genutzten `config` Datei angeben. Wenn das SDK die erforderliche Konfiguration nicht auflösen kann, können API-Anfragen zur Laufzeit fehlschlagen.

Neben dieser allgemeinen Konfigurationskette verwendet AWS SDK für PHP Version 3 auch spezialisierte Anbieterketten, darunter die [Anbieterkette für Anmeldeinformationen](#) und die [AWS-Region Auflösungskette](#). Diese spezialisierten Ketten fügen zusätzliche Anbieter hinzu, die die Umgebung berücksichtigen, in der das SDK ausgeführt wird. Zum Beispiel in einem Container oder einer EC2 Instanz.

Erstellen Sie einen Service-Client, der mit externen Einstellungen konfiguriert wurde

Sie müssen in Ihrer Anwendung einen Service-Client erstellen, um mit einem zu kommunizieren AWS-Service. Serviceclients sind Ihre unverzichtbare Verbindung zu AWS-Services ihnen. Sie kümmern sich um alle komplexen Kommunikationsdetails, sodass Sie sich keine Gedanken über sie machen müssen. Sie kümmern sich automatisch um wichtige Aufgaben wie Sicherheit, Fehlerbehandlung und Wiederholungen, sodass Sie sich auf die Erstellung Ihrer Anwendung konzentrieren können, anstatt sich um technische Komplikationen zu kümmern.

Konstruieren Sie einen Service-Client mit einem Konstruktor ohne Parameter

Wenn alle benötigten Konfigurationseinstellungen aus externen Quellen stammen, können Sie einen Dienstclient erstellen, indem Sie einen leeren Konstruktor verwenden:

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

use Aws\S3\S3Client;

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

Der vorherige Codeausschnitt erstellt eine Instanz. `S3Client` Während der Erstellung durchsucht das SDK die Kette der Konfigurationsanbieter nach Einstellungen. Sobald das SDK einen Einstellungswert gefunden hat, verwendet es den Wert.

Die AWS-Region Standardanbieterkette und die Standard-Anbieterkette für Anmeldeinformationen werden ebenfalls im Erstellungsprozess verwendet. Irgendwo in der Kette muss das SDK die AWS-Region zu verwendenden Einstellungen auflösen und Einstellungen finden, die es ermöglichen, Anmeldeinformationen für Signieranfragen abzurufen. Wenn das SDK diese Werte nicht findet, schlägt die Client-Erstellung fehl.

AWS SDK für PHP Umgebungsvariablen der Version 3

Neben den [SDK-übergreifenden Einstellungen](#), die von den meisten unterstützt werden AWS SDKs, arbeitet AWS SDK für PHP Version 3 mit den folgenden Umgebungsvariablen:

AWS_SDK_LOAD_NONDEFAULT_CONFIG

Wenn diese Umgebungsvariable gesetzt ist, weist sie das SDK an, zusätzlich zu den Datei(en) auch Anmeldeinformationen aus der AWS config Datei (~/.aws/config) zu laden.

`credentials ~/.aws/credential`

AWS_SDK_UA_APP_ID

Legt eine benutzerdefinierte Anwendungs-ID fest, die in den User-Agent-Header für Anfragen des SDK aufgenommen wird.

AWS_SUPPRESS_PHP_DEPRECATION_WARNING

Wenn dieser Wert auf true gesetzt ist, werden Warnungen vor PHP-Verwarnungen unterdrückt, die das SDK möglicherweise generiert.

Konfiguration von Service-Clients im Code für die AWS SDK für PHP Version 3

Als Alternative oder zusätzlich zur [externen Konfiguration von Service-Clients](#) können Sie sie programmgesteuert im Code konfigurieren.

Durch die Konfiguration von Service-Clients im Code erhalten Sie eine detaillierte Kontrolle über die vielen Optionen, die Ihnen zur Verfügung stehen. Die meisten Konfigurationen, die Sie extern festlegen können, stehen Ihnen auch zur Verfügung, um sie im Code festzulegen.

Grundkonfiguration im Code

Sie können einen Service-Client im Code erstellen und konfigurieren, indem Sie ein assoziatives Array von Optionen an den Konstruktor eines Clients übergeben. Im folgenden Beispiel enthält das assoziative Array nur die Option „Region“, die der Client verwendet:

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

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

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

```
]);
```

Informationen zum optionalen Parameter „Version“ finden Sie im Thema [Konfigurationsoptionen](#).

Beachten Sie, dass wir nicht explizit Anmeldeinformationen für den Client angegeben haben. Das liegt daran, dass das SDK die [standardmäßige Anbieterkette für Anmeldeinformationen](#) verwendet, um nach Anmeldeinformationen zu suchen.

Alle allgemeinen Client-Konfigurationsoptionen werden ausführlich beschrieben. [Client-Konstruktor-Optionen für die AWS SDK für PHP Version 3](#) Die Anzahl der Optionen, die einem Client zur Verfügung gestellt werden, kann je nach Client, den Sie erstellen, variieren. Diese benutzerdefinierten Clientkonfigurationsoptionen sind in der [API-Dokumentation](#) für jeden Client beschrieben.

Verwendung der **Sdk** Klasse

Die Klasse `Aws\Sdk` fungiert als Client-Factory und wird verwendet, um gemeinsame Konfigurationsoptionen für mehrere Clients zu verwalten. Viele der Optionen, die einem bestimmten Client-Konstruktor zur Verfügung gestellt werden können, können auch der `Aws\Sdk` Klasse zur Verfügung gestellt werden. Diese Optionen werden dann auf jeden Client-Konstruktor angewendet.

Importe

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

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

Beispiel-Code

```
// 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();
```

Optionen, die für alle Clients freigegeben sind, werden in Schlüssel/Wert-Paaren auf Stammebene platziert. Dienstspezifische Konfigurationsdaten können in einem assoziativen Array mit einem Schlüssel bereitgestellt werden, der dem Namespace eines Dienstes entspricht (z. B. „S3“, „DynamoDb“ usw.).

```
$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();
```

Servicespezifische Konfigurationswerte sind eine Vereinigung der servicespezifischen Werte und der Werte auf Stammebene (d. h. servicespezifische Werte werden flach auf Werte auf Stammebene zusammengeführt).

Note

Wir empfehlen dringend, dass Sie die Klasse Sdk verwenden, um Clients zu erstellen, wenn Sie mehrere Client-Instances in Ihrer Anwendung verwenden. Die Klasse Sdk verwendet automatisch denselben HTTP-Client für jeden SDK-Client, sodass SDK-Clients für verschiedene Services nicht blockierende HTTP-Anforderungen ausführen können. Wenn die SDK-Clients nicht denselben HTTP-Client verwenden, blockieren HTTP-Anforderungen, die vom SDK-Client gesendet werden, möglicherweise die Promise-Orchestrierung zwischen Services.

Client-Konstruktor-Optionen für die AWS SDK für PHP Version 3

Client-Konstruktor-Optionen können in einem Client-Konstruktor oder der [Aws\Sdk](#)-Klasse bereitgestellt werden. Die Anzahl der Optionen, die einem bestimmten Clienttyp zur Verfügung stehen, kann variieren, je nachdem, welchen Client Sie anlegen. Diese benutzerdefinierten Client-Konfigurationsoptionen sind in der [API-Dokumentation](#) für jeden Client beschrieben.

Wenn Sie die vom Client benötigten Client-Konstruktoroptionen nicht explizit angeben, sucht das SDK for PHP nach Werten aus Umgebungsvariablen oder aus einer AWS Konfigurationsdatei. Alle Clients benötigen einen Credential Provider-Wert und einen AWS-Region Wert, daher müssen Sie diese Werte als Konstruktoroptionen oder extern festlegen.

Standardmäßig befindet sich die zu überprüfende Konfigurationsdatei `.aws/config` in Ihrem Stammverzeichnis, in der Regel `~/.aws/config`. Sie können jedoch die Umgebungsvariable `AWS_CONFIG_FILE` verwenden, um den Standardspeicherort der Konfigurationsdatei festzulegen. Dies kann beispielsweise nützlich sein, wenn Sie den Dateizugriff auf bestimmte Verzeichnisse mit einschränken. `open_basedir`

Weitere Informationen zum Speicherort und zur Formatierung der gemeinsam genutzten AWS `config` `credentials` Dateien finden Sie unter [Konfiguration](#) im Referenzhandbuch für Tools AWS SDKs und Tools.

Einzelheiten zu allen globalen Konfigurationseinstellungen, die Sie in den AWS Konfigurationsdateien oder als Umgebungsvariablen festlegen können, finden Sie in der Referenz zu den [Einstellungen für Konfiguration AWS SDKs und Authentifizierung im Referenzhandbuch](#) für Tools.

Konfigurationsoptionen

- [api_provider](#)
- [Anmeldedaten](#)
- [debug](#)
- [stats](#)
- [Endpunkt](#)
- [endpoint_provider](#)
- [endpoint_discovery](#)
- [handler](#)
- [http](#)
- [http_handler](#)
- [Profil](#)
- [Region](#)
- [retries](#)
- [scheme](#)

- [Service nicht zulässig](#)
- [signature_provider](#)
- [signature_version](#)
- [ua_append](#)
- [use_aws_shared_config_files](#)
- [validieren](#)
- [version](#)

Das folgende Beispiel zeigt, wie Optionen an einen Amazon S3 S3-Client-Konstruktor übergeben werden.

```
use Aws\S3\S3Client;

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

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

Weitere Informationen [the section called “Einen Basis-Serviceclient erstellen”](#) zum Konstruieren von Clients finden Sie unter.

api_provider

Typ

callable

Eine aufrufbare PHP-Funktion, die ein Typ-, Service- und Versionsargument entgegennimmt und ein Array mit entsprechenden Konfigurationsdaten zurückgibt. Der Wert für den Typ kann `api`, `waiter` oder `paginator` sein.

Standardmäßig verwendet das SDK eine Instance von `Aws\Api\FileSystemApiProvider`, die Dateien aus dem `src/data`-Ordner der SDK-API lädt.

Anmeldedaten

Typ

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

Übergeben Sie ein `Aws\Credentials\CredentialsInterface`-Objekt, um eine spezifische Anmeldeinformationen-Instance zu verwenden. Im Folgenden wird angegeben, dass der IAM Identity Center-Anmeldeinformationsanbieter verwendet werden sollte. Dieser Anbieter wird auch als SSO-Anmeldeinformationsanbieter bezeichnet.

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

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

Wenn Sie ein benanntes Profil verwenden, ersetzen Sie 'default' im vorherigen Beispiel durch den Namen Ihres Profils. Weitere Informationen zum Einrichten benannter Profile finden Sie unter [Geteilte Profile config und credentials Dateien](#) im AWS SDKs Referenzhandbuch zu Tools.

Wenn Sie keinen zu verwendenden Anmeldeinformationsanbieter angeben und sich auf die Anmeldeinformationsanbieterkette verlassen, ist die Fehlermeldung aufgrund einer fehlgeschlagenen Authentifizierung in der Regel generisch. Sie wird anhand des letzten Anbieters in der Liste der Quellen generiert, die auf gültige Anmeldeinformationen überprüft werden. Dabei handelt es sich möglicherweise nicht um den Anbieter, den Sie verwenden möchten. Wenn Sie angeben, welcher Anmeldeinformationsanbieter verwendet werden soll, ist jede daraus resultierende Fehlermeldung hilfreicher und relevanter, da sie nur von diesem Anbieter stammt. Weitere Informationen zur Kette von Quellen, die auf Anmeldeinformationen überprüft wurden, finden Sie unter [Credential Provider Chain](#) im AWS SDKs Tools-Referenzhandbuch.

Übergeben Sie `false`, um keine Anmeldeinformationen zu verwenden und Anfragen nicht zu signieren.

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

```
]);
```

Übergeben Sie eine aufrufbare [Anmeldeinformationsanbieter](#)-Funktion, um Anmeldeinformationen unter Verwendung einer Funktion zu erstellen.

```
use Aws\Credentials\CredentialProvider;

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

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

Übergeben Sie in einer Instance von `Aws\CacheInterface` zwischengespeicherte Anmeldeinformationen, um die Werte von der Standard-Anbieterkette über mehrere Prozesse zwischenzuspeichern.

```
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
]);
```

Weitere Informationen zur Bereitstellung von Anmeldeinformationen für einen Client finden Sie im Handbuch [Anmeldeinformationen für AWS SDK für PHP Version 3](#).

Note

Anmeldeinformationen werden langsam geladen und geprüft, wenn sie verwendet werden.

debug

Typ

`bool|array`

Gibt Debugging-Informationen zu jeder Übertragung aus. Debugging-Informationen enthalten Informationen zu jeder Statusänderung einer Transaktion, wie sie erstellt und gesendet wird. Außerdem sind in der Debugging-Ausgabe Informationen zum jeweiligen HTTP-Handler enthalten, der von einem Client verwendet wird (z. B. debug cURL-Ausgabe).

Auf `true` setzen, um Debugging-Informationen anzuzeigen, wenn Anforderungen gesendet werden.

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

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

Alternativ können Sie eine assoziatives Array mit den folgenden Schlüsseln bereitstellen.

logfn (callable)

Funktion, die mit Protokolleinträgen aufgerufen wird. Standardmäßig wird die echo-Funktion von PHP verwendet.

stream_size (int)

Wenn die Größe eines Datenstroms größer als diese Zahl ist, werden die Stream-Daten nicht protokolliert. Auf `0` setzen, um nicht alle Stream-Daten zu protokollieren.

scrub_auth (bool)

Stellen Sie diese Option auf `false`, um das Löschen von Authentifizierungsdaten aus den protokollierten Nachrichten zu deaktivieren (was bedeutet, dass Ihre AWS Zugriffsschlüssel-ID und Signatur an die `logfn` weitergegeben werden).

http (bool)

Auf `false` setzen, um die „Debug“-Funktion von HTTP-Handlern auf niedrigerer Ebene zu deaktivieren (z. B. verbose cURL-Ausgabe).

auth_headers (array)

Auf eine Schlüssel-Wert-Zuweisung von Headern setzen, die Sie ersetzen wollen, abgebildet auf den Wert, durch den Sie sie ersetzen möchten. Diese Werte werden nicht verwendet, es sei denn `scrub_auth` ist auf `true` gesetzt.

auth_strings (array)

Auf eine Schlüssel-Wert-Zuordnung regulärer Ausdrücke setzen, um eine Abbildung auf ihre Ersatzwerte vorzunehmen. Diese Werte werden vom Authentifizierungsdaten-Scrubber verwendet, wenn `scrub_auth` auf `true` gesetzt ist.

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

Diese Option gibt auch die zugrunde liegenden HTTP-Handler-Informationen aus, die von der `http Debug`-Option erzeugt wurden. Die Debugging-Ausgabe ist extrem hilfreich beim Diagnostizieren von Problemen in AWS SDK für PHP. Bitte geben Sie die Debugging-Ausgabe für einen isolierten Fehlerfall an, wenn Sie Tickets für das SDK eröffnen.

stats

Typ

`bool|array`

Bindet Übertragungsstatistiken an Fehler und Ergebnisse, die von SDK-Operationen zurückgegeben wurden.

Auf `true` setzen, um Übertragungsstatistiken für gesendete Anforderungen zu erfassen.

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

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

Alternativ können Sie eine assoziatives Array mit den folgenden Schlüsseln bereitstellen.

retries (bool)

Auf `true` einstellen, `false` um die Berichterstattung über wiederholte Versuche zu deaktivieren. Wiederholungsstatistiken werden standardmäßig erfasst und zurückgegeben.

http (bool)

Wird auf `true` gesetzt, `true` um das Sammeln von Statistiken von HTTP-Adaptern auf niedrigerer Ebene zu ermöglichen (z. B. zurückgegebene Werte). GuzzleHttpTransferStats HTTP-Handler müssen eine `__on_transfer_stats`-Option unterstützen, damit dies eine Wirkung zeigt. HTTP-Statistiken werden als ein indiziertes Array assoziativer Arrays zurückgegeben; jedes assoziative Array enthält die Übertragungsstatistiken, die der HTTP-Handler des Clients für eine Anfrage zurückgibt. Standardmäßig deaktiviert.

Falls eine Anforderung wiederholt wurde, werden die Übertragungsstatistiken für jede Anforderung zurückgegeben, wobei `$result['@metadata']['transferStats']['http'][0]` die Statistiken für die erste Anforderung, `$result['@metadata']['transferStats']['http'][1]` die Statistiken für die zweite Anforderung und so weiter enthält.

timer (bool)

Auf `true` setzen, um einen Befehlstimer zu aktivieren, der die gesamte für eine Operation aufgewendete Zeit in Sekunden anzeigt. Standardmäßig deaktiviert.

```
$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

Typ

string

Die vollständige URI des Webservice. Dies ist beispielsweise für Dienste erforderlich, die [AWS Elemental MediaConvert](#) kontospezifische Endpunkte verwenden. Für diese Dienste fordern Sie diesen Endpunkt mit der Methode `describeEndpoints` an.

Dies ist nur erforderlich, wenn Sie eine Verbindung zu einem benutzerdefinierten Endpunkt herstellen (z. B. eine lokale Version von Amazon S3 oder [Amazon DynamoDB Local](#)).

Hier ist ein Beispiel für eine Verbindung zu Amazon DynamoDB Local:

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region' => 'us-east-1',
```



```
'endpoint' => 'http://localhost:8000'  
]);
```

Eine Liste der verfügbaren [AWS Regionen und Endpunkte](#) finden Sie unter AWS Regionen und Endpunkte.

endpoint_provider

Typ

`Aws\EndpointV2\EndpointProviderV2|callable`

Eine optionale aufrufbare Instanz von EndpointProvider V2 oder PHP, die einen Hash von Optionen akzeptiert, einschließlich eines „Service“- und „Region“-Schlüssels. Sie gibt NULL oder einen Hash für Endpunktdaten zurück, von denen der „Endpunktschlüssel“ benötigt wird.

Hier folgt ein Beispiel dafür, wie ein minimaler Endpunktanbieter eingerichtet wird.

```
$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

Typ

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

Die Endpunkterkennung identifiziert den korrekten Endpunkt für Service-APIs, die die Endpunkterkennung unterstützen, und stellt eine Verbindung damit her. Aktivieren Sie während der Client-Erstellung `endpoint_discovery` für Services, die die Endpunkterkennung zwar unterstützen, aber nicht erfordern. Wenn ein Service die Endpunkterkennung nicht unterstützt, wird diese Konfiguration ignoriert.

`Aws\EndpointDiscovery\ConfigurationInterface`

Ein optionaler Konfigurationsanbieter, der automatische Verbindungsherstellung zum entsprechenden Endpunkt einer Service-API für Operationen ermöglicht, die der Service bestimmt.

Das Objekt `Aws\EndpointDiscovery\Configuration` akzeptiert zwei Optionen, darunter ein Boolescher Wert („enabled“), der angibt, ob die Endpunkterkennung aktiviert ist, und eine Ganzzahl („cache_limit“), die die maximale Anzahl von Schlüsseln im Endpunkt-Cache angibt.

Für jeden erstellten Client übergeben Sie ein Objekt `Aws\EndpointDiscovery\Configuration`, um eine bestimmte Konfiguration für die Endpunkterstellung zu nutzen.

```
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,
]);
```

Übergeben Sie eine Instance von `Aws\CacheInterface`, um die Werte von der Endpunkterkennung über mehrere Prozesse im Cache zu speichern.

```
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),
]);
```

Übergeben Sie der Endpunkterkennung ein Array.

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

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

handler

Typ

callable

Ein Handler, der ein Befehlsobjekt und ein Anfrageobjekt akzeptiert und ein Versprechen zurückgibt (GuzzleHttp\Promise\PromiseInterface), das mit einem -Objekt erfüllt oder mit einer Aws\ResultInterface abgelehnt wird. Ein Handler akzeptiert keinen weiteren Handler, da er ein Terminal ist und einen Befehl ausführen soll. Wenn kein Handler bereitgestellt wird, wird eine Standard-Guzzle-Handler verwendet.

Du kannst den Aws\MockHandler benutzen, um modellhafte Ergebnisse zurückzugeben oder modellhafte Ausnahmen aufzuwerfen. Sie stellen Ergebnisse oder Ausnahmen in die Warteschlange und MockHandler dann werden sie in der FIFO-Reihenfolge aus der Warteschlange entfernt.

```
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

Typ

array

Auf ein Array von HTTP-Optionen setzen, die auf vom SDK erstellte HTTP-Anforderungen und -Übertragungen angewendet werden.

Das SDK unterstützt die folgenden Konfigurationsoptionen:

cert

Typ

string|array

Geben Sie das PEM-formatierte Client-seitige Zertifikat an.

- Als Zeichenfolge für den Pfad nur zur Zertifikatdatei festlegen.

```
use Aws\S3\S3Client;

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

- Legen Sie dies als Array fest, das Pfad und Passwort enthält.

```
use Aws\S3\S3Client;

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

connect_timeout

Ein Fließkommawert, der die Anzahl der Sekunden angibt, die man warten muss, während man versucht, eine Verbindung zu einem Server herzustellen. Verwenden Sie 0 für unbestimmtes Warten (das Standardverhalten).

```
use Aws\DynamoDb\DynamoDbClient;

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

debug

Typ

`bool|resource`

Weist den zugrundeliegenden HTTP-Handler an, Debugging-Informationen auszugeben. Die Debugging-Informationen von verschiedenen HTTP-Handlern variieren.

- Übergeben Sie `true`, um Debugging-Ausgaben auf STDOUT zu schreiben.
- Übergeben Sie eine `resource` wie von `fopen` zurückgegeben, um Debugging-Ausgabe auf eine spezifische PHP-Stream-Ressource zu schreiben.

decode_content

Typ

bool

Weist den zugrundeliegenden HTTP-Handler an, den Rumpf der komprimierten Antworten zu erweitern. Wenn dies nicht aktiviert ist, werden komprimierte Antwortrumpfe möglicherweise mit einem `GuzzleHttp\Psr7\InflateStream` erweitert.

Note

Die Inhaltsdekodierung ist im Standard-HTTP-Handler des SDK standardmäßig aktiviert. Aus Gründen der Abwärtskompatibilität kann diese Voreinstellung nicht geändert werden. Wenn Sie komprimierte Dateien in Amazon S3 speichern, empfehlen wir, die Inhaltsdekodierung auf S3-Client-Ebene zu deaktivieren.

```
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
```

Verzögerung

Typ

int

Die Anzahl der Millisekunden für die Verzögerung vor dem Senden der Anfrage. Dies wird oft verwendet, um eine Anfrage zu verzögern, bevor sie erneut versucht wird.

`expect`

Typ

`bool|string`

Diese Option wird durch die zugrunde liegenden HTTP-Handler übergeben. Standardmäßig werden Kopfzeilen mit „Expect: 100 Continue“ festgelegt, wenn der Textteil der Anforderung 1 MB überschreitet. `true` oder `false` aktiviert oder deaktiviert die Kopfzeile auf allen Anforderungen. Wenn eine Ganzzahl verwendet wird, verwenden nur Anforderungen diese Kopfzeile, deren Textteil diese Einstellung überschreitet. Bei Verwendung als Ganzzahl wird die expect-Kopfzeile mitgesendet, wenn die Textgröße nicht bekannt ist.

 Warning

Die Deaktivierung der expect-Kopfzeile kann Fehler hervorrufen, z. B. kann es sein, dass der Service keine Authentifizierung mehr zurückgibt. Diese sollte mit Vorsicht konfiguriert werden.

`progress`

Typ

`callable`

Definiert eine Funktion, die aufgerufen wird, wenn Übertragungsfortschritt gemacht wird. Die Funktion akzeptiert die folgenden Argumente:

1. Die Gesamtanzahl der Bytes, die voraussichtlich heruntergeladen werden.
2. Die Anzahl der Bytes, die bisher heruntergeladen wurden.
3. Die Gesamtanzahl der Bytes, die voraussichtlich hochgeladen werden.
4. Die Anzahl der Bytes, die bisher hochgeladen wurden.

```
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

Typ

string|array

Mithilfe der proxy Option können Sie über einen Proxy eine Verbindung zu einem AWS Service herstellen.

- Geben Sie einen Zeichenfolgenwert an, um eine Verbindung zu einem Proxy für alle Typen von herzustellen URIs. Der Proxy-Zeichenfolgenwert kann ein Schema, einen Benutzernamen und ein Passwort enthalten. Beispiel, "http://username:password@192.168.16.1:10".
- Stellen Sie ein assoziatives Array mit Proxy-Einstellungen bereit, wobei der Schlüssel das Schema der URI ist, und der Wert der Proxy für die angegebene URI (d. h. Sie können verschiedene Proxys für „http“- und „https“-Endpunkte angeben).

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

Sie können die `HTTP_PROXY`-Umgebungsvariable verwenden, sodass ein „http“-Protokoll-spezifischer Proxy konfiguriert wird, und die `HTTPS_PROXY`-Umgebungsvariable, sodass ein „https“-spezifischer Proxy konfiguriert wird.

sink

Typ

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

Die `sink`-Option steuert, wohin die Antwortdaten einer Operation heruntergeladen werden.

- Geben Sie eine `resource` wie von `fopen` zurückgegeben zum Herunterladen des Antworttrumpfs in einen PHP-Stream an.
- Geben Sie den Pfad zu einer Datei auf der Festplatte als `string`-Wert an, um den Antworttrumpf in eine bestimmte Datei auf der Festplatte herunterzuladen.
- Geben Sie ein `Psr\Http\Message\StreamInterface` an, um den Antworttrumpf in ein spezifisches PSR-Streamobjekt herunterzuladen.

 Note

Das SDK lädt den Antwortrumpf standardmäßig in einen temporären PHP-Stream herunter. Das bedeutet, dass die Daten im Speicher bleiben, bis die Größe des Rumpfs 2 MB erreicht. Zu diesem Zeitpunkt werden die Daten in eine temporäre Datei auf der Festplatte geschrieben.

synchronous

Typ

bool

Die `synchronous`-Option informiert den zugrundeliegenden HTTP-Handler darüber, dass Sie das Ergebnis blockieren wollen.

stream

Typ

bool

Auf `true` setzen, um dem zugrundeliegenden HTTP-Handler mitzuteilen, dass Sie den Antwortrumpf einer Antwort vom Webservice streamen möchten, anstatt alles im Voraus herunterzuladen. Diese Option wird beispielsweise in der Amazon S3 `S3-Stream-Wrapper`-Klasse verwendet, um sicherzustellen, dass die Daten gestreamt werden.

timeout

Typ

float

Eine Fließkommazahl, die das Timeout der Anfrage in Sekunden beschreibt. Verwenden Sie `0` für unbestimmtes Warten (das Standardverhalten).

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

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

verify

Typ

bool|string

Mithilfe der Option können Sie das Verhalten des SDK bei der Überprüfung von SSL/TLS Peer-Zertifikaten anpassen. `verify http`

- Stellen Sie diese `true` Option ein, um die Überprüfung von SSL/TLS Peer-Zertifikaten zu aktivieren und das vom Betriebssystem bereitgestellte Standard-CA-Paket zu verwenden.
- Auf `false` setzen, um die Peer-Zertifikatsverifizierung zu deaktivieren. (Das ist nicht sicher!)
- Auf eine Zeichenfolge setzen, um den Pfad zu einem CA-Zertifikat-Bundle bereitzustellen, um die Verifizierung mit einem benutzerdefinierten CA-Bundle zu ermöglichen.

Wenn das CA-Bundle für Ihr System nicht gefunden werden kann und Sie einen Fehler erhalten, stellen Sie dem SDK den Pfad zu einem CA-Bundle bereit. Wenn Sie kein bestimmtes CA-Bundle benötigen, bietet Mozilla ein häufig verwendetes CA-Bundle an, das Sie [hier](#) herunterladen können (dies wird vom Betreuer von cURL gepflegt.). Sobald Sie ein CA-Bundle auf der Festplatte haben, können Sie die `openssl.cafile` PHP .ini so einstellen, dass sie auf den Pfad zur Datei verweist, sodass Sie die `verify`-Anfrageoption weglassen können. Weitere Informationen über SSL-Zertifikate finden Sie auf der [cURL-Website](#).

```
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

Typ

callable

Die `http_handler`-Option wird für die Integration des SDK mit anderen HTTP-Clients verwendet. Eine `http_handler`-Option ist eine Funktion, die ein `Psr\Http\Message\RequestInterface`-Objekt akzeptiert, und ein Array mit `http`-Optionen, die auf den Befehl angewendet werden, und die ein `GuzzleHttp\Promise\PromiseInterface`-Objekt zurückgibt, das mit einem `Psr\Http\Message\ResponseInterface`-Objekt erfüllt oder mit einem Array mit den folgenden Ausnahmedaten abgelehnt wird:

- `exception` – (`\Exception`) die Ausnahme, die festgestellt wurde.
- `response` – (`Psr\Http\Message\ResponseInterface`) die (gegebenenfalls) erhaltene Antwort.
- `connection_error` – (`bool`) gleich `true`, um den Fehler als Verbindungsfehler zu kennzeichnen. Wenn dieser Wert auf `true` gesetzt wird, kann das SDK die Operation gegebenenfalls automatisch wiederholen.

Das SDK konvertiert automatisch den vorgegebenen `http_handler` in eine normale `handler`-Option, indem es den bereitgestellten `http_handler` in ein `Aws\WrappedHttpHandler`-Objekt kapselt.

Standardmäßig verwendet das SDK Guzzle als HTTP-Handler. Sie können hier einen anderen HTTP-Handler angeben oder einen Guzzle-Client mit eigenen benutzerdefinierten Optionen bereitstellen.

Festlegen der TLS-Version

Ein Anwendungsfall besteht darin, die von Guzzle verwendete TLS-Version mit Curl festzulegen, vorausgesetzt, dass Curl in Ihrer Umgebung installiert ist. Beachten Sie die Curl-[Versionseinschränkungen](#) für die unterstützte Version von TLS. Standardmäßig wird die neueste Version verwendet. Wenn die TLS-Version explizit festgelegt ist und der Remoteserver diese Version nicht unterstützt, wird anstelle einer früheren TLS-Version ein Fehler ausgegeben.

Sie können die TLS-Version festlegen, die für eine bestimmte Client-Operation verwendet wird, indem Sie die debug-Client-Option auf „true“ setzen und die SSL-Verbindungs Ausgabe prüfen. Diese Zeile könnte etwa so aussehen: `SSL connection using TLSv1.2`

Beispiel für das Festlegen von TLS 1.2 mit 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

Die `http_handler`-Option ersetzt alle angegebenen `handler`-Optionen.

Profil

Typ

`string`

Die Option „Profil“ gibt an, welches Profil verwendet werden soll, wenn Anmeldeinformationen aus der AWS Anmeldeinformationsdatei in Ihrem HOME-Verzeichnis erstellt werden (normalerweise `~/.aws/credentials`). Diese Einstellung überschreibt die `AWS_PROFILE`-Umgebungsvariable.

Note

Wenn Sie die Option „Profil“ angeben, wird die `"credentials"` Option ignoriert und die Einstellungen für Anmeldeinformationen in der AWS Konfigurationsdatei werden (normalerweise `~/.aws/config`) ignoriert.

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

Weitere Informationen [zur Konfiguration von Anmeldeinformationen und zum .ini-Dateiformat finden Sie unter Anmeldeinformationen für AWS SDK für PHP Version 3.](#)

Region

Typ

string

Erforderlich

true

AWS Region, zu der eine Verbindung hergestellt werden soll. Eine Liste der verfügbaren [AWS Regionen finden Sie unter Regionen und Endpunkte.](#)

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

retries

Typ

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

Standard

`int(3)`

Konfiguriert den Wiederholungsmodus und die maximal zulässige Anzahl von Wiederholungen für einen Client. Übergeben Sie `0`, um Wiederholungen zu deaktivieren.

Die drei Wiederholungsmodi sind:

- `legacy`- die standardmäßige Legacy-Wiederholungsimplementierung
- `standard`- fügt ein Quotensystem für Wiederholungsversuche hinzu, um Wiederholungsversuche zu verhindern, bei denen es unwahrscheinlich ist, dass sie erfolgreich sind
- `adaptive` – baut auf dem Standardmodus auf und fügt eine clientseitige Ratenbegrenzung hinzu. Beachten Sie, dass dieser Modus als experimentell betrachtet wird.

Die Konfiguration für Wiederholungen besteht aus dem Modus und den maximal zulässigen Versuchen für die einzelnen Anforderungen. Die Konfiguration kann an mehreren verschiedenen Orten in der folgenden Rangfolge festgelegt werden.

Rangfolge

Die Rangfolge für die Wiederholungskonfiguration lautet wie folgt (1 überschreibt 2-3 usw.):

1. Client-Konfigurationsoption
2. Umgebungsvariablen
3. AWS Gemeinsam genutzte Konfigurationsdatei

Umgebungsvariablen

- `AWS_RETRY_MODE` – festgelegt auf `legacy`, `standard` oder `adaptive`
- `AWS_MAX_ATTEMPTS` – festgelegt auf einen ganzzahligen Wert für die maximal zulässigen Versuche pro Anforderung

Schlüssel für Datei mit gemeinsam verwendeter Konfiguration

- `retry_mode` – festgelegt auf `legacy`, `standard` oder `adaptive`
- `max_attempts` – festgelegt auf einen ganzzahligen Wert für die maximal zulässigen Versuche pro Anforderung

Client-Konfiguration

Das folgende Beispiel deaktiviert Wiederholungen für den Amazon DynamoDB-Client.

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

Im folgenden Beispiel wird ein ganzzahliger Wert übergeben, der standardmäßig den `legacy`-Modus mit der übergebenen Anzahl von Wiederholungen aufweist

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

Das Objekt **`Aws\Retry\Configuration`** akzeptiert zwei Parameter, den Wiederholungsmodus

und einen ganze Wert für die maximal zulässigen Versuche pro Anforderung. In diesem Beispiel wird ein

`Aws\Retry\Configuration`-Objekt für die Wiederholungskonfiguration übergeben.

```
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,
]);
```

In diesem Beispiel wird ein Array für die Wiederholungskonfiguration übergeben.

```
use Aws\S3\S3Client;

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

In diesem Beispiel wird eine Instance von `Aws\CacheInterface` übergeben, um die vom Standardanbieter für Wiederholungskonfigurationen zurückgegebenen Werte zwischenspeichern.

```
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

Typ

string

Standard

string(5) "https"

URI-Schema, das für die Verbindung verwendet werden soll. Das SDK verwendet standardmäßig „https“-Endpunkte (d. h., SSL/TLS-Verbindungen). Sie können versuchen, eine Verbindung zu einem Service über einen unverschlüsselten „http“-Endpunkt herzustellen, indem Sie `scheme` auf „http“ setzen.

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

Unter [AWS Regionen und Endpunkte finden Sie eine Liste der Endpunkte](#) und ob ein Service das Schema unterstützt. `http`

Service nicht zulässig

Typ

`string`

Erforderlich

`true`

Name des zu verwendenden Service. Dieser Wert wird standardmäßig bereitgestellt, wenn Sie einen vom SDK bereitgestellten Client verwenden (z. B. `Aws\S3\S3Client`). Diese Option ist nützlich, wenn Sie einen Service testen, der noch nicht im SDK veröffentlicht wurde, aber auf der Festplatte verfügbar ist.

`signature_provider`

Typ

`callable`

Ein Callable, das einen Signaturversionsnamen (z. B. `v4`), einen Dienstenamen und eine AWS Region akzeptiert und ein `Aws\Signature\SignatureInterface` Objekt zurückgibt oder NULL, ob der Anbieter in der Lage ist, einen Unterzeichner für die angegebenen Parameter zu erstellen. Dieser Anbieter wird verwendet, um vom Client verwendete Signaturgeber zu erstellen.

Es gibt verschiedene Funktionen des SDK in der `Aws\Signature\SignatureProvider`-Klasse, die genutzt werden können, um benutzerdefinierte Signaturanbieter zu erstellen.

`signature_version`

Typ

`string`

Eine Zeichenfolge, die eine benutzerdefinierte Signaturversion darstellt, die für einen Service verwendet wird (z. B. `v4` usw.). Die pro Operation verwendete Signaturversion KANN diese angeforderte Signaturversion bei Bedarf übersteuern.

Die folgenden Beispiele zeigen, wie ein Amazon S3 S3-Client für die Verwendung der [Signaturversion 4](#) konfiguriert wird:

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

Note

Der von Ihrem Client verwendete `signature_provider` MUSS die Möglichkeit bieten, die von Ihnen bereitgestellte `signature_version`-Option zu erstellen. Der vom SDK verwendete standardmäßige `signature_provider` kann Signaturobjekte für „v4“ und „anonyme“ Signaturversionen erstellen.

`ua_append`

Typ

`string|string[]`

Standard

`[]`

Eine Zeichenfolge oder ein Array von Zeichenfolgen, die der dem HTTP-Handler übergebenen Benutzeragenten-Zeichenfolge hinzugefügt werden.

`use_aws_shared_config_files`

Typ

`bool|array`

Standard

`bool(true)`

Auf `false` setzen, um die Suche nach einer gemeinsam genutzten Konfigurationsdatei in `'~/aws/config'` und `'~/aws/credentials'`. Dadurch wird die `AWS_CONFIG_FILE` Umgebungsvariable überschrieben.

`validieren`

Typ

`bool|array`

Standard

`bool(true)`

Auf `false` setzen, um die clientseitige Parametervalidierung zu deaktivieren. Sie werden vielleicht feststellen, dass die Deaktivierung der Validierung die Leistung des Clients etwas verbessert, aber der Unterschied ist vernachlässigbar.

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

Auf ein assoziatives Array von Validierungsoptionen setzen, um spezifische Validierungsbedingungen zu aktivieren:

- `required` – Validieren, ob die erforderlichen Parameter vorhanden sind (standardmäßig aktiviert).
- `min` – Validieren der Mindestlänge eines Werts (standardmäßig aktiviert).
- `max` – Validieren der maximalen Länge eines Werts.
- `pattern` – Validieren, ob der Wert mit einem regulären Ausdruck übereinstimmt.

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

version

Typ

`string`

Erforderlich

`false`

Diese Option gibt die Version des Webdienstes an, die verwendet werden soll (z. B. `2006-03-01`).

Ab Version 3.277.10 des SDK ist die Option „Version“ nicht erforderlich. Wenn Sie die Option „Version“ nicht angeben, verwendet das SDK die neueste Version des Service-Clients.

In zwei Situationen ist ein „Version“-Parameter erforderlich, wenn Sie einen Service-Client erstellen.

- Sie verwenden eine ältere Version des PHP-SDK als 3.277.10.
- Sie verwenden Version 3.277.10 oder höher und möchten eine andere Version als die „neueste“ für einen Service-Client verwenden.

Das folgende Snippet verwendet beispielsweise Version 3.279.7 des SDK, aber nicht die neueste Version für `Ec2Client`

```
$ec2Client = new \Aws\Ec2\Ec2Client([
```

```
'version' => '2015-10-01',  
'region' => 'us-west-2'  
]);
```

Die Angabe einer Versionsbeschränkung stellt sicher, dass Ihr Code nicht durch eine Unterbrechung des Service beeinträchtigt wird.

Eine Liste der verfügbaren API-Versionen finden Sie auf den Seiten für die [API-Dokumentation](#) für jeden Client. Wenn Sie eine bestimmte API-Version nicht laden können, müssen Sie möglicherweise Ihre Kopie des SDK aktualisieren.

Einstellung der AWS-Region für die AWS SDK für PHP Version 3

SDK-Clients stellen eine Verbindung zu einem bestimmten AWS-Service in einem bestimmten Bereich der AWS-Region, den Sie bei der Erstellung des Clients angeben. Diese Konfiguration ermöglicht es Ihrer Anwendung, mit AWS Ressourcen in diesem geografischen Gebiet zu interagieren. Wenn Sie einen Service-Client erstellen, ohne explizit eine Region festzulegen, verwendet das SDK die Standardregion aus Ihrer externen Konfiguration.

Auflösungskette für Regionen

AWS SDK für PHP Version 3 verwendet die folgende Reihenfolge, um zu bestimmen, welche Region ein Service-Client verwendet:

1. Im Code angegebene Region — Wenn Sie die Region in den Client-Konstruktoroptionen explizit festlegen, hat dies Vorrang vor allen anderen Quellen.

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

2. Umgebungsvariablen — Wenn im Code keine Region angegeben ist, sucht das SDK der Reihe nach nach diesen Umgebungsvariablen:

- AWS_REGION
- AWS_DEFAULT_REGION

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

3. AWS Konfigurationsdateien — Wenn keine Umgebungsvariablen für die Region festgelegt sind, überprüft das SDK die Konfigurationsdateien: AWS
 - a. Das SDK sucht in `~/.aws/config` (oder in dem in der `AWS_CONFIG_FILE` Umgebungsvariablen angegebenen Speicherort)
 - b. Das SDK untersucht die Regionseinstellung innerhalb des durch die `AWS_PROFILE` Umgebungsvariable angegebenen Profils
 - c. Wenn kein Profil angegeben `AWS_PROFILE` ist, verwendet das SDK das „Standard“-Profil

Nehmen wir als Beispiel an, dass wir die folgenden Einstellungen für die Konfigurationsdatei haben:

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

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

Wenn für die `AWS_PROFILE` Umgebungsvariable der Wert „production“ festgelegt ist, verwenden Clients `eu-central-1` Region. Wenn keine `AWS_PROFILE` Umgebungsvariable vorhanden ist, verwenden Clients die `eu-west-1` Region.

4. Wenn das SDK in keiner der oben genannten Quellen einen Regionswert findet, löst es eine Ausnahme aus, da ein Regionswert eine erforderliche Einstellung für einen Service-Client ist.

Bewährte Methoden

Beachten Sie bei der Arbeit mit Regionen in AWS SDK für PHP Version 3 die folgenden bewährten Methoden:

Legen Sie die Region explizit im Produktionscode fest

Für Produktionsanwendungen empfehlen wir, die Region explizit in Ihrem Code festzulegen, anstatt sich auf Umgebungsvariablen oder die zu verlassen `config`. Dadurch wird Ihr Code vorhersehbarer und ist weniger von der externen Konfiguration abhängig.

Verwenden Sie Umgebungsvariablen für Entwicklung und Tests

In Entwicklungs- und Testumgebungen ermöglicht die Verwendung von Umgebungsvariablen mehr Flexibilität, ohne den Code zu ändern.

Verwenden Sie Profile für mehrere Umgebungen

Wenn Ihre Anwendung mit mehreren AWS Umgebungen arbeiten muss, sollten Sie erwägen, unterschiedliche Profile in Ihrer AWS config Datei zu verwenden und bei Bedarf zwischen diesen zu wechseln.

Verwenden von Anmeldeinformationsanbietern der AWS SDK für PHP Version 3

Referenzinformationen zu den verfügbaren Anmeldemechanismen für finden Sie unter [Anmeldeinformationen und Zugriff](#) im AWS SDKs Referenzhandbuch zu Tools. AWS SDKs

Important

Aus Sicherheitsgründen empfehlen wir dringend, nicht das Root-Konto für den AWS Zugriff zu verwenden. Aktuelle [Sicherheitsempfehlungen finden Sie immer unter Bewährte Sicherheitsmethoden in IAM](#) im IAM-Benutzerhandbuch.

Die Rolle eines Anbieters für Anmeldeinformationen in AWS SDK für PHP Version 3 besteht darin, Anmeldeinformationen für die SDK-Clients zu beschaffen und bereitzustellen. AWS-Service Das SDK verwendet die Anmeldeinformationen, die es bezieht, um sich beim Dienst zu authentifizieren, indem es jede Anfrage kryptografisch signiert. Anmeldeinformationen bestehen normalerweise aus Zugriffsschlüsseln — einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel zusammen.

Wenn Sie temporäre Anmeldeinformationen verwenden, z. B. wenn Sie die [IAM Identity Center-Authentifizierung einrichten](#) oder Ihre Runtime so konfigurieren, dass sie [eine IAM-Rolle übernimmt](#), wird den Zugriffsschlüsseln ein Sitzungstoken hinzugefügt, das einen zeitlich begrenzten Zugriff auf Ressourcen ermöglicht. AWS

Was ist ein Anmeldeinformationsanbieter in Version 3? AWS SDK für PHP

Ein Anmeldeinformationsanbieter ist eine Funktion, die eine `GuzzleHttp\Promise\PromiseInterface` zurückgibt, die mit einer `Aws\Credentials\CredentialsInterface`-Instance erfüllt oder mit einer `Aws\Exception\CredentialsException` abgelehnt wird. Das [SDK bietet verschiedene Implementierungen](#) von Funktionen des Anmeldeinformationsanbieters. Sie können aber auch [Ihre eigene benutzerdefinierte Logik zum Erstellen von Anmeldeinformationen oder zum Optimieren des Ladens von Anmeldeinformationen implementieren](#).

Anmeldeinformationsanbieter werden in der `credentials`-Option des Client-Konstruktors übergeben. Anmeldeinformationsanbieter sind asynchron, sodass sie jedes Mal, wenn eine API-Operation aufgerufen wird, langsam ausgewertet werden. Die Übergabe einer Anmeldeinformationsanbieter-Funktion an einen SDK-Client-Konstruktor führt daher nicht sofort zur Validierung der Anmeldeinformationen. Wenn der Anmeldeinformationsanbieter kein Anmeldeinformationsobjekt zurückgibt, wird eine API-Operation mit einer `Aws\Exception\CredentialsException` abgelehnt.

```
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
]);
```

Grundlegendes zur Standard-Anbieterkette für Anmeldeinformationen in AWS SDK für PHP Version 3

Die standardmäßige Anbieterkette für Anmeldeinformationen besteht aus einer Reihe integrierter Anmeldeinformationsanbieter, die das SDK aufruft. Es wird durch die [DefaultProvider-Funktion](#) `DefaultProvider` für Anmeldeinformationen ohne Parameter implementiert. Nachdem gültige Anmeldeinformationen gefunden wurden, wird die Suche beendet.

Der AWS SDK für PHP führt Anmeldeinformationsanbieter in der folgenden Reihenfolge aus:

- [envprovider](#) — Das SDK sucht nach [AWS Zugriffsschlüsseln, die als Umgebungsvariablen festgelegt wurden](#).
- [assumeRoleWithWebIdentityCredentialProviderprovider](#) — Das SDK sucht nach Dateieinstellungen für die IAM-Rolle und das Web-Identitätstoken.
- An diesem Punkt in der Kette sucht das SDK nach Konfigurationen in den gemeinsam genutzten `credentials` Dateien AWS config und Dateien. Das SDK sucht unter dem „Standard“-Profil

nach der Konfiguration, aber wenn die `AWS_PROFILE` Umgebungsvariable gesetzt ist, verwendet das SDK ihren benannten Profilwert.

- [`ssoprovider`](#) — Das SDK sucht in der gemeinsam genutzten `config` Datei nach den [IAM Identity Center-Konfigurationseinstellungen](#).
- [`login provider`](#) - Das SDK sucht in der gemeinsam genutzten `config` Datei nach Konfigurationseinstellungen für die AWS Konsolen-Anmeldesitzung.
- [`processAnbieter`](#) — Das SDK sucht in der gemeinsam genutzten `credentials` Datei nach der `credential_process` Einstellung.
- [`iniproducer`](#) — Das SDK sucht in der gemeinsam genutzten `credentials` Datei nach den AWS Anmeldeinformationen oder IAM-Rolleninformationen.
- [`processprovider`](#) — Das SDK sucht in der gemeinsam genutzten `config` Datei nach der `credential_process` Einstellung.
- [`iniproducer`](#) — Das SDK sucht in der gemeinsam genutzten `config` Datei nach den AWS Anmeldeinformationen oder IAM-Rolleninformationen.
- [`ecsCredentialsprovider`](#) — Das SDK sucht nach Umgebungsvariablen `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` oder Variablen `AWS_CONTAINER_CREDENTIALS_FULL_URI`, die Informationen zum Abrufen temporärer Anmeldeinformationen bereitstellen.
- [`instanceProfileprovider`](#) — Das SDK verwendet den EC2 Instanz-Metadatendienst, um die im Instanzprofil angegebene IAM-Rolle abzurufen. Mithilfe der Rolleninformationen ruft das SDK temporäre Anmeldeinformationen ab.

Note

Das Ergebnis des Standard-Anbieters wird automatisch gespeichert.

Sie können den Code für die Kette im GitHub [Quellcode](#) überprüfen.

Integrierte Anmeldeinformationsanbieter in der AWS SDK für PHP Version 3

Das SDK bietet mehrere integrierte Anbieter von Anmeldeinformationen, die Sie einzeln verwenden oder in einer [benutzerdefinierten](#) Anbieterkette kombinieren können.

Wenn Sie bei der Erstellung des Dienstclients einen Anbieter für Anmeldeinformationen angeben, versucht das SDK, Anmeldeinformationen zu laden, indem es nur den angegebenen Anmeldeinformationsanbieter verwendet. Es verwendet nicht die [standardmäßige Anbieterkette für Anmeldeinformationen](#). Wenn Sie wissen, dass ein Dienstclient den [instanceProfile](#) Anbieter verwenden soll, können Sie die Standardkette kurzschließen, indem Sie den `instanceProfile` Anbieter im Service-Client-Konstruktor angeben:

```
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

Anmeldeinformationsanbieter werden jedes Mal aufgerufen, wenn eine API-Operation ausgeführt wird. Wenn das Laden von Anmeldeinformationen eine teure Aufgabe ist (z. B. das Laden von Festplatten oder eine Netzwerkressource), oder wenn Anmeldeinformationen von Ihrem Anbieter nicht im Cache gespeichert werden, sollten Sie in Betracht ziehen, Ihren Anmeldeinformationsanbieter in eine `Aws\Credentials\CredentialProvider::memoize`-Funktion zu verpacken. Das SDK merkt sich den Standard-Anmeldeinformationsanbieter automatisch.

Themen

- [loginAnbieter im SDK for PHP](#)
- [assumeRoleAnbieter im SDK for PHP](#)
- [ssoAnbieter im SDK for PHP](#)
- [defaultProviderAnbieter im SDK for PHP](#)
- [ecsCredentialsAnbieter im SDK for PHP](#)

- [envAnbieter im SDK for PHP](#)
- [assumeRoleWithWebIdentityCredentialProviderAnbieter im SDK for PHP](#)
- [iniAnbieter im SDK for PHP](#)
- [processAnbieter im SDK for PHP](#)
- [instanceProfileAnbieter im SDK for PHP](#)

loginAnbieter im SDK for PHP

Aws\Credentials\CredentialProvider::loginversucht, Anmeldeinformationen zu laden, die durch eine browserbasierte Anmeldesitzung konfiguriert wurden, die durch Tools wie die AWS CLI unterstützt wird. AWS Generiert nach der Authentifizierung temporäre Anmeldeinformationen, die lokal AWS SDKs und in allen Tools funktionieren.

Mit diesem Prozess können Sie sich mit Root-Anmeldeinformationen, die bei der ersten Kontoeinrichtung erstellt wurden, mit einem IAM-Benutzer oder einer föderierten Identität von Ihrem Identitätsanbieter authentifizieren. Das AWS SDK for PHP verwaltet die temporären Anmeldeinformationen automatisch für Sie. Dieser Ansatz erhöht die Sicherheit, da die Notwendigkeit entfällt, langfristige Anmeldeinformationen lokal zu speichern.

Wenn Sie den `aws login` Befehl ausführen, können Sie aus Ihren aktiven Konsolensitzungen wählen oder sich über den browserbasierten Authentifizierungsablauf anmelden. Dabei werden automatisch temporäre Anmeldeinformationen generiert. Das AWS SDK for PHP aktualisiert diese Anmeldeinformationen mithilfe des Anmeldedienstes automatisch für bis zu 12 Stunden.

Der Anmeldeanbieter versucht, das durch den zuvor erwähnten Ablauf der Anmeldesitzung generierte Zugriffstoken auf der Grundlage des bereitgestellten Profils zu laden. Wenn beim Aufrufen des Anbieters kein Profil angegeben wird, versucht dieser, ein Profil aufzulösen, indem zuerst die `AWS_PROFILE` Umgebungsvariable überprüft wird, bevor auf das Profil zurückgegriffen wird `default`. Die Code-Konfiguration kann an den Anbieter übergeben werden, der dann nach einem `region` Wert für den Anmeldedienstclient sucht, der zum Aktualisieren der Anmeldeinformationen verwendet wird. Wenn im Konfigurationsarray keine Region angegeben ist, versucht der Anbieter, eine Region aufzulösen, indem er die `AWS_REGION` Umgebungsvariable und dann einen im aufgelösten Profil festgelegten Regionswert überprüft. Wenn keine Region gefunden werden kann, gibt der Anbieter eine abgelehnte Zusage mit Anweisungen zur Konfiguration einer Region zurück.

Der Anbieter wird als Teil der Standardkette aufgerufen und kann direkt aufgerufen werden.

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

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

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

Wenn auf dem Service-Client, den Sie verwenden möchten, keine Konfiguration für Anmeldeinformationen bereitgestellt wird, wird dieser Anbieter standardmäßig als Teil der `defaultProvider()` Anmeldeinformationskette aufgerufen. In diesem Szenario wird die Region des Service-Clients automatisch an den `login()` Anbieter übergeben. Auch in diesem Szenario wird der an den Anmeldeanbieter übergebene Profilwert aufgelöst, indem die `AWS_PROFILE` Umgebungsvariable überprüft wird, bevor auf das Profil zurückgegriffen wird `default`.

assumeRoleAnbieter im SDK for PHP

Wenn Sie `Aws\Credentials\AssumeRoleCredentialProvider` zum Erstellen von Anmeldeinformationen verwenden, indem Sie eine Rolle annehmen, müssen Sie `'client'`-Informationen mit einem `StsClient`-Objekt und `'assume_role_params'`-Details bereitstellen, wie dargestellt.

Note

Um zu vermeiden, dass bei jedem API-Vorgang AWS STS Anmeldeinformationen unnötig abgerufen werden, können Sie die `memoize` Funktion verwenden, um die Anmeldeinformationen automatisch zu aktualisieren, wenn sie ablaufen. Nachfolgend finden Sie ein Codebeispiel.

```
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
]);
```

Weitere Informationen 'assume_role_params' dazu finden Sie unter. [AssumeRole](#)

ssoAnbieter im SDK for PHP

`Aws\Credentials\CredentialProvider::sso` ist der Anbieter von Single Sign-On-Anmeldeinformationen. Dieser Anbieter wird auch als AWS IAM Identity Center Anmeldeinformationsanbieter bezeichnet.

```
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
]);
```

Wenn Sie ein benanntes Profil verwenden, ersetzen Sie 'default' im vorherigen Beispiel durch den Namen Ihres Profils. Weitere Informationen zum Einrichten benannter Profile finden Sie unter [Geteilte Profile config und credentials Dateien](#) im AWS SDKs Referenzhandbuch zu Tools. Alternativ können Sie die [AWS_PROFILE](#) Umgebungsvariable verwenden, um anzugeben, welche Profileinstellungen verwendet werden sollen.

Weitere Informationen zur Funktionsweise des IAM Identity Center-Anbieters finden Sie unter [Grundlegendes zur IAM Identity Center-Authentifizierung](#) im AWS SDKs Referenzhandbuch zu Tools.

defaultProvider Anbieter im SDK for PHP

`Aws\Credentials\CredentialProvider::defaultProvider` ist der Standardanbieter für Anmeldeinformationen und wird auch als [Standardanbieterkette für Anmeldeinformationen](#) bezeichnet. Dieser Anbieter wird verwendet, wenn Sie bei der Erstellung eines Clients die `credentials`-Option nicht angeben. Wenn Sie beispielsweise einen `S3Client` erstellen, wie im folgenden Codeausschnitt gezeigt, verwendet das SDK den Standardanbieter:

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

Sie können den `defaultProvider` auch im Code verwenden, wenn Sie bestimmten Anmeldeinformationsanbietern in der Kette Parameter zur Verfügung stellen möchten. Das folgende Beispiel bietet beispielsweise benutzerdefinierte Einstellungen für Verbindungstimeout und Wiederholungsversuche, wenn die `ecsCredentials` Provider-Funktion verwendet wird.

```
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
]);
```

ecsCredentialsAnbieter im SDK for PHP

`Aws\Credentials\CredentialProvider::ecsCredentials` versucht, Anmeldeinformationen über eine GET-Anforderung zu laden, deren URI durch die Umgebungsvariable `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` im Container angegeben ist.

```
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
]);
```

envAnbieter im SDK for PHP

Die Verwendung von Umgebungsvariablen zur Speicherung Ihrer Anmeldeinformationen verhindert, dass Sie versehentlich Ihren AWS geheimen Zugriffsschlüssel weitergeben. Wir empfehlen, dass Sie Ihre AWS Zugangsschlüssel niemals direkt dem Client in Produktionsdateien hinzufügen.

Um sich bei Amazon Web Services zu authentifizieren, sucht das SDK zunächst nach Anmeldeinformationen in Ihren Umgebungsvariablen. Das SDK verwendet die `getenv()`-Funktion, um nach den Umgebungsvariablen `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` und `AWS_SESSION_TOKEN` zu suchen. Diese Anmeldeinformationen werden als Umgebungs-Anmeldeinformationen bezeichnet. Anweisungen zum Abrufen dieser Werte finden Sie unter [Authentifizieren mit kurzfristigen Anmeldeinformationen](#) im Referenzhandbuch AWS SDKs zu Tools.

Wenn Sie Ihre Anwendung auf [AWS Elastic Beanstalk](#) hosten, können Sie die `AWS_SESSION_TOKEN` Umgebungsvariablen `AWS_ACCESS_KEY_ID`, `AWS_SECRET_KEY`, und [über die AWS Elastic Beanstalk Konsole](#) festlegen, sodass das SDK diese Anmeldeinformationen automatisch verwenden kann.

Weitere Informationen zum Einstellen von Umgebungsvariablen finden Sie unter [Unterstützung von Umgebungsvariablen](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. Eine Liste aller Umgebungsvariablen, die von den meisten unterstützt werden AWS SDKs, finden Sie auch unter [Liste der Umgebungsvariablen](#).

Sie können die Umgebungsvariablen auch in der Befehlszeile festlegen, wie hier gezeigt.

Linux

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

Windows

```
C:\> SET  AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
# The access key for your AWS-Konto.
C:\> SET  AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
# The secret access key for your AWS-Konto.
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
# The temporary session key for your AWS-Konto.
# 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` versucht, Anmeldeinformationen aus Umgebungsvariablen zu laden.

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

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

```
]);
```

assumeRoleWithWebIdentityCredentialProvider Anbieter im SDK for PHP

Aws\Credentials

\CredentialProvider::assumeRoleWithWebIdentityCredentialProvider versucht Anmeldeinformationen durch Übernehmen einer Rolle zu laden. Wenn die Umgebungsvariablen `AWS_ROLE_ARN` und `AWS_WEB_IDENTITY_TOKEN_FILE` vorhanden sind, versucht der Anbieter, die mit `AWS_ROLE_ARN` angegebene Rolle unter Verwendung des Tokens auf dem Datenträger in dem mit `AWS_WEB_IDENTITY_TOKEN_FILE` angegebenen vollständigen Pfad zu übernehmen. Wenn Umgebungsvariablen verwendet werden, versucht der Anbieter, die Sitzung mit der Umgebungsvariablen `AWS_ROLE_SESSION_NAME` einzurichten.

Wenn keine Umgebungsvariablen festgelegt wurden, verwendet der Anbieter das Standardprofil oder das als `AWS_PROFILE` festgelegte Profil. Der Anbieter liest Profile standardmäßig in `~/.aws/credentials` und `~/.aws/config` und kann Profile lesen, die in der Konfigurationsoption `filename` angegeben sind. Der Anbieter übernimmt die in `role_arn` des Profils angegebene Rolle, indem ein Token in dem in `web_identity_token_file` angegebenen vollständigen Pfad gelesen wird. `role_session_name` wird verwendet, wenn dies im Profil festgelegt ist.

Der Anbieter wird als Teil der Standardkette aufgerufen und kann direkt aufgerufen werden.

```
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
]);
```

Standardmäßig erbt dieser Anmeldeinformationsanbieter die konfigurierte Region, die von dem verwendet wird, `StsClient` um die Rolle zu übernehmen. Optional kann eine vollständige Datei bereitgestellt `StsClient` werden. Die Anmeldeinformationen sollten wie `false` bei allen angegebenen angegeben werden `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
]);
```

iniAnbieter im SDK for PHP

`Aws\Credentials\CredentialProvider::ini` versucht, Anmeldeinformationen aus den geteilten Dateien `config` und `credentials` Dateien zu laden. Standardmäßig versucht das SDK, das „Standard“-Profil aus der gemeinsam genutzten AWS `credentials` Datei zu laden, die sich unter `~/.aws/credentials` befindet. Wenn das SDK die `AWS_SDK_LOAD_NONDEFAULT_CONFIG` Umgebungsvariable findet, sucht es auch nach einem „Standard“-Profil in der gemeinsam genutzten AWS `config` Datei unter `~/.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  
]);
```

Sie können ein benutzerdefiniertes Profil oder einen .ini-Dateispeicherort verwenden, indem Sie der Funktion Argumente übergeben, die den Anbieter erstellt.

```
$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  
]);
```

processAnbieter im SDK for PHP

Aws\Credentials\CredentialProvider::processversucht, Anmeldeinformationen zu laden, indem ein credential_process Wert ausgeführt wird, der in einem Profil in einer [gemeinsam genutzten AWS Konfigurationsdatei](#) angegeben ist.

Standardmäßig versucht das SDK, das „Standard“-Profil zuerst aus der gemeinsam genutzten AWS credentials Datei zu laden, die sich unter befindet ~/ .aws/credentials. Wenn das „Standard“-Profil in der gemeinsam genutzten credentials Datei nicht gefunden wird, sucht das SDK in der gemeinsam genutzten config Datei nach dem Standardprofil. Im Folgenden finden Sie ein Beispiel für die Konfiguration der gemeinsam genutzten credentials Datei.

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

Das SDK ruft den credential_process Befehl genau so auf, wie er mithilfe der shell_exec PHP-Funktion angegeben wurde, und liest dann JSON-Daten aus stdout. Sie credential_process müssen Anmeldeinformationen im folgenden Format auf stdout schreiben:

```
{  
    "Version": 1,
```

```
"AccessKeyId": "",
"SecretAccessKey": "",
"SessionToken": "",
"Expiration": ""
}
```

`SessionToken` und `Expiration` sind optional. Falls vorhanden, werden die Anmeldeinformationen als temporär behandelt.

```
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
]);
```

Sie können ein benutzerdefiniertes Profil oder einen .ini-Dateispeicherort verwenden, indem Sie der Funktion Argumente übergeben, die den Anbieter erstellt.

```
$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
]);
```

instanceProfileAnbieter im SDK for PHP

`Aws\Credentials\CredentialProvider::instanceProfile` versucht, Anmeldeinformationen für eine IAM-Rolle zu laden, die in einem EC2 Amazon-Instance-Profil angegeben ist.

```
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
]);
```

Standardmäßig versucht der Anbieter bis zu drei Mal, die Anmeldeinformationen abzurufen. Die Anzahl der Wiederholungen kann mit der `retries` Option festgelegt und vollständig deaktiviert werden, indem die Option auf gesetzt wird, `0` wie im folgenden Code gezeigt.

```
use Aws\Credentials\CredentialProvider;

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

Wenn die Umgebungsvariable verfügbar `AWS_METADATA_SERVICE_NUM_ATTEMPTS` ist, hat ihr Wert Vorrang vor der zuvor gezeigten Option „Wiederholungen“.

Note

Sie können diesen Versuch, aus EC2 Amazon-Instance-Profilen zu laden, deaktivieren, indem Sie die `AWS_EC2_METADATA_DISABLED` Umgebungsvariable auf `set>true`.

Verkettung von Anmeldeinformationsanbietern im SDK for PHP

Sie können Anmeldeinformationsanbieter mit der Funktion `Aws\Credentials\CredentialProvider::chain()` verketteten. Diese Funktion akzeptiert eine variable Anzahl von Argumenten, wobei es sich jeweils um Anmeldeinformationsanbieter-Funktionen handelt. Diese Funktion gibt dann eine neue Funktion zurück, die eine Zusammensetzung der bereitgestellten

Funktionen ist, sodass sie nacheinander aufgerufen werden, bis einer der Anbieter ein Promise zurückgibt, das erfolgreich erfüllt wurde.

Der `defaultProvider` verwendet diese Zusammenstellung, um mehrere Anbieter zu überprüfen, bevor ein Fehler zurückgegeben wird. Die Quelle des `defaultProvider` demonstriert die Nutzung der `chain`-Funktion.

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

Erstellen eines benutzerdefinierten Anmeldeinformationsanbieters zur Verwendung mit dem SDK for PHP

Anmeldeinformationsanbieter sind einfach Funktionen, die beim Aufrufen ein Promise (`GuzzleHttp\Promise\PromiseInterface`) zurückgeben, das mit einem `Aws\Credentials\CredentialsInterface`-Objekt erfüllt oder mit einer `Aws\Exception\CredentialsException` abgelehnt wird.

Eine gute Vorgehensweise zum Erstellen von Anbietern ist es, eine Funktion zu erstellen, die aufgerufen wird, um den eigentlichen Anmeldeinformationsanbieter zu erstellen. Als Beispiel finden Sie hier die Quelle des `env`-Anbieters (für Beispielzwecke leicht abgeändert). Beachten Sie, dass es sich um eine Funktion handelt, die die eigentliche Anbieter-Funktion zurückgibt. Auf diese Weise können Sie problemlos die Anmeldeinformationsanbieter erstellen und sie als Werte übergeben.

```
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));
    };
}

```

Speichern von Anmeldeinformationen im SDK for PHP

Manchmal ist es notwendig, einen Anmeldeinformationsanbieter zu erstellen, der sich den vorherigen Rückgabewert merkt. Dies kann für die Performance nützlich sein, wenn das Laden von Anmeldeinformationen eine teure Operation ist oder wenn die Klasse `Aws\Sdk` verwendet wird, um einen Anmeldeinformationsanbieter für mehrere Clients freizugeben. Sie können einem Anmeldeinformationsanbieter das Speichern hinzufügen, indem Sie die Anmeldeinformationsanbieter-Funktion in eine memoize-Funktion verpacken.

```

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());
```

Wenn die gespeicherten Anmeldeinformationen abgelaufen sind, ruft der Memorize-Wrapper den verpackten Anbieter auf, um die Anmeldeinformationen zu aktualisieren.

Nehmen Sie eine IAM-Rolle mit AWS SDK für PHP Version 3 an

Verwenden von IAM-Rollen für EC2 Amazon-Instance-Variablenanmeldedaten

Wenn Sie Ihre Anwendung auf einer EC2 Amazon-Instance ausführen, besteht die bevorzugte Methode zur Bereitstellung von Anmeldeinformationen für Aufrufe AWS darin, eine [IAM-Rolle](#) zu verwenden, um temporäre Sicherheitsanmeldeinformationen abzurufen.

Wenn Sie IAM-Rollen verwenden, müssen Sie sich keine Gedanken über die Verwaltung der Anmeldeinformationen in Ihrer Anwendung machen. Sie ermöglichen es einer Instance, eine Rolle zu „übernehmen“, indem sie temporäre Anmeldeinformationen vom Metadatenserver der EC2 Amazon-Instance abrufen.

Die temporären Anmeldeinformationen, die oft als Anmeldeinformationen für das Instance-Profil bezeichnet werden, ermöglichen den Zugriff auf die Aktionen und Ressourcen, die die Rollenrichtlinie zulässt. Amazon EC2 kümmert sich um die sichere Authentifizierung von Instances gegenüber dem IAM-Service, um die Rolle zu übernehmen, und um die regelmäßige Aktualisierung der abgerufenen Rollenanmeldedaten. Damit bleibt Ihre Anwendung sicher, ohne dass Sie selbst etwas tun müssen. Eine Liste der Dienste, die temporäre Sicherheitsanmeldedaten akzeptieren, finden Sie im IAM-Benutzerhandbuch unter [AWS Dienste, die mit IAM funktionieren](#).

Note

Um zu vermeiden, dass jedes Mal der Metadaten-Service benötigt wird, kann eine Instance von `Aws\CacheInterface` als `'credentials'`-Option an einen Client-Konstruktor übergeben werden. Auf diese Weise kann das SDK stattdessen im Cache gespeicherte Instance-Profil-Anmeldeinformationen verwenden. Einzelheiten finden Sie unter [Konfiguration für AWS SDK für PHP Version 3](#).

Weitere Informationen zur Entwicklung von EC2 Amazon-Anwendungen mithilfe von finden Sie unter [Verwenden von IAM-Rollen für EC2 Amazon-Instances](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. SDKs

Eine IAM-Rolle erstellen und einer EC2 Amazon-Instance zuweisen

1. Erstellen Sie einen IAM-Client.

Importe

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

use Aws\Iam\IamClient;
```

Beispiel-Code

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

2. Erstellen Sie eine IAM-Rolle mit den Berechtigungen für die Aktionen und Ressourcen, die Sie verwenden werden.

Beispiel-Code

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

3. Erstellen Sie ein IAM-Instance-Profil und speichern Sie den Amazon-Ressourcennamen (ARN) aus dem Ergebnis.

Note

Wenn Sie die IAM-Konsole anstelle von verwenden AWS SDK für PHP, erstellt die Konsole automatisch ein Instance-Profil und weist diesem denselben Namen zu wie der Rolle, der es entspricht.

Beispiel-Code

```
$IPN = 'InstanceProfileName';

$result = $client->createInstanceProfile([
```

```
'InstanceProfileName' => $IPN ,
]);

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

4. Erstellen Sie einen EC2 Amazon-Client.

Importe

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

use Aws\Ec2\Ec2Client;
```

Beispiel-Code

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

5. Fügen Sie das Instance-Profil zu einer laufenden oder angehaltenen EC2 Amazon-Instance hinzu. Verwenden Sie den Instance-Profilnamen Ihrer IAM-Rolle.

Beispiel-Code

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

Weitere Informationen finden Sie unter [IAM-Rollen für Amazon EC2](#) im EC2 Amazon-Benutzerhandbuch.

Verwenden von IAM-Rollen für Amazon ECS-Aufgaben

Eine Aufgabe in Amazon Elastic Container Service (Amazon ECS) kann eine IAM-Rolle für AWS API-Aufrufe übernehmen. Dies ist eine Strategie zur Verwaltung der Anmeldeinformationen, die Ihre Anwendungen verwenden sollen, ähnlich wie EC2 Amazon-Instance-Profile Anmeldeinformationen für EC2 Amazon-Instances bereitstellen.

Anstatt langfristige AWS Anmeldeinformationen zu erstellen und an Container zu verteilen oder die Rolle der EC2 Amazon-Instance zu verwenden, können Sie eine IAM-Rolle, die temporäre Anmeldeinformationen verwendet, mit einer ECS-Aufgabendefinition oder einem RunTask [API-Vorgang](#) verknüpfen.

Weitere Informationen zur Verwendung von IAM-Rollen, die Container-Tasks annehmen können, finden Sie im Thema [Task-IAM-Rolle](#) im Amazon ECS Developer Guide. Beispiele für die Verwendung der Task-IAM-Rolle in Form einer `taskRoleArn` in Aufgabendefinitionen finden Sie unter [Beispielaufgabendefinitionen](#) ebenfalls im Amazon ECS Developer Guide.

Übernahme einer IAM-Rolle in einer anderen AWS-Konto

Wenn Sie in einem AWS-Konto (Konto A) arbeiten und eine Rolle in einem anderen Konto (Konto B) übernehmen möchten, müssen Sie zunächst eine IAM-Rolle in Konto B erstellen. Diese Rolle ermöglicht es Entitäten in Ihrem Konto (Konto A), bestimmte Aktionen in Konto B durchzuführen. Weitere Informationen zum kontoübergreifenden Zugriff finden Sie unter [Tutorial: Kontoübergreifendes Delegieren des Zugriffs mithilfe von IAM-Rollen AWS](#).

Notieren Sie sich nach dem Erstellen der Rolle in Konto B den ARN. Sie verwenden diesen ARN, wenn Sie die Rolle von Konto A übernehmen. Sie übernehmen die Rolle mit den AWS Anmeldeinformationen, die Ihrer Entität in Konto A zugeordnet sind.

Erstellen Sie einen AWS STS Client mit Anmeldeinformationen für Ihren AWS-Konto. Im Folgenden wird hierzu ein Anmeldeinformationsprofil verwendet, aber Sie können eine beliebige Methode nutzen. Rufen Sie mit dem neu erstellten AWS STS -Client `assume-role` auf und legen Sie einen benutzerdefinierten `sessionName` fest. Rufen Sie die neuen temporären Anmeldeinformationen aus dem Ergebnis ab. Die Anmeldeinformationen dauern standardmäßig eine Stunde.

Beispiel-Code

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

Weitere Informationen finden Sie unter [Verwenden von IAM-Rollen](#) oder [AssumeRole](#) in der AWS SDK für PHP API-Referenz.

Verwenden einer IAM-Rolle mit Webidentität

Web Identity Federation ermöglicht es Kunden, beim Zugriff auf AWS Ressourcen externe Identitätsanbieter für die Authentifizierung zu verwenden. Bevor Sie eine Rolle mit Web-Identität übennehmen können, müssen Sie zunächst eine IAM-Rolle erstellen und einen Web-Identitätsanbieter (Identity provider, IdP) konfigurieren. Weitere Informationen finden Sie unter [Erstellen von Rollen für Web-Identität oder OpenID Connect-Verbund \(Konsole\)](#).

Nachdem Sie [einen Identitätsanbieter](#) und [eine Rolle für Ihre Web-Identität erstellt](#) haben, verwenden Sie einen AWS STS Client, um einen Benutzer zu authentifizieren. Geben Sie das `webIdentityToken` und `ProviderId` für Ihre Identität und den Rollen-ARN für die IAM-Rolle mit Berechtigungen für den Benutzer an.

Beispiel-Code

```
$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']
    ]
]);

```

Weitere Informationen finden Sie unter [AssumeRoleWithWebIdentity—Verbund über einen webbasierten Identitätsanbieter](#) oder [AssumeRoleWithWebIdentity](#) in der AWS SDK für PHP API-Referenz.

Nehmen Sie die Rolle mit dem Profil an

Definieren Sie Profile in `~/.aws/credentials`

Sie können die AWS SDK für PHP für die Verwendung einer IAM-Rolle konfigurieren, indem Sie ein Profil in `~/.aws/credentials` definieren.

Erstellen Sie ein neues Profil mit der `role_arn` Einstellung für die Rolle, die Sie übernehmen möchten. Fügen Sie auch die `source_profile` Einstellung für ein anderes Profil mit Anmeldeinformationen hinzu, die berechtigt sind, die IAM-Rolle anzunehmen. Weitere Informationen zu diesen Konfigurationseinstellungen finden Sie unter [Rollenanmeldedaten annehmen](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Im Folgenden legt das Profil `~/.aws/credentials` beispielsweise das `project1` Profil fest `role_arn` und gibt das `default` Profil als Quelle für Anmeldeinformationen an, um zu überprüfen, ob die mit ihnen verknüpfte Entität die Rolle übernehmen kann.

```
[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
```

Wenn Sie die `AWS_PROFILE` Umgebungsvariable festlegen oder bei der Instanziierung eines Dienstclients `profile` Parameter verwenden, wird die in angegebene Rolle übernommen, wobei das `default` Profil als Quellenmeldeinformationen verwendet `project1` wird.

Der folgende Ausschnitt zeigt die Verwendung des `profile` Parameters in einem Konstruktor. `S3Client` `S3Client` Sie werden über die Berechtigungen verfügen, die der Rolle zugeordnet sind, die dem Profil zugeordnet ist. `project1`

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

Definieren Sie Profile in `~/.aws/config`

Die `~/.aws/config` Datei kann auch Profile enthalten, von denen Sie annehmen möchten. Wenn Sie die Umgebungsvariable setzen `AWS_SDK_LOAD_NONDEFAULT_CONFIG`, lädt das SDK for PHP Profile aus der `config` Datei. Wenn `AWS_SDK_LOAD_NONDEFAULT_CONFIG` diese Option gesetzt ist, lädt das SDK Profile `~/.aws/config` sowohl aus als auch `~/.aws/credentials`. Profile von `~/.aws/credentials` werden zuletzt geladen und haben Vorrang vor Profilen von `~/.aws/config` mit demselben Namen. Profile aus beiden Speicherorten können als `source_profile` oder als das zu übernehmende Profil dienen.

Im folgenden Beispiel werden das in der config Datei definierte `project1` Profil und das `default` Profil in der `credentials` Datei verwendet. Das `AWS_SDK_LOAD_NONDEFAULT_CONFIG` ist ebenfalls festgelegt.

```
# 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
```

Wenn der `S3Client` Konstruktor ausgeführt wird, der im folgenden Codeausschnitt gezeigt wird, wird die im Profil definierte Rolle anhand der mit dem `project1` Profil verknüpften Anmeldeinformationen übernommen. `default`

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

Verwenden Sie temporäre Anmeldeinformationen aus AWS STS dem SDK for PHP

AWS -Security-Token-Service (AWS STS) ermöglicht es Ihnen, eingeschränkte Rechte und temporäre Anmeldeinformationen für IAM-Benutzer oder für Benutzer, die Sie über einen Identitätsverbund authentifizieren, anzufordern. Weitere Informationen finden Sie unter [Temporäre Sicherheitsanmeldedaten](#) im IAM-Benutzerhandbuch. Für den Zugriff auf die meisten AWS Dienste können Sie temporäre Sicherheitsanmeldedaten verwenden. Eine Liste der Dienste, die temporäre Sicherheitsanmeldedaten akzeptieren, finden Sie im [IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren](#).

Ein häufiger Anwendungsfall für temporäre Anmeldeinformationen besteht darin, mobilen oder clientseitigen Anwendungen Zugriff auf AWS Ressourcen zu gewähren, indem Benutzer über externe Identitätsanbieter authentifiziert werden (siehe Web Identity Federation).

Abrufen temporärer Anmeldeinformationen

AWS STS hat mehrere Operationen, die temporäre Anmeldeinformationen zurückgeben, aber der `GetSessionToken` Vorgang ist am einfachsten zu demonstrieren. Das folgende Snippet ruft temporäre Anmeldeinformationen ab, indem es die `getSessionToken` Methode des STS-Clients des PHP-SDK aufruft.

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

$stsClient = $sdk->createSts();

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

Das Ergebnis für `GetSessionToken` und die anderen AWS STS Operationen enthält immer einen Wert. `'Credentials'` Wenn Sie das drucken `$result` (zum Beispiel mit `print_r($result)`), sieht es wie folgt aus.

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

Bereitstellung temporärer Anmeldeinformationen für AWS SDK für PHP

Sie können temporäre Anmeldeinformationen mit einem anderen AWS Client verwenden, indem Sie den Client instanziiieren und die von AWS STS diesem empfangenen Werte direkt übergeben.

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

Sie können auch ein `Aws\Credentials\Credentials`-Objekt konstruieren und es bei der Instanziierung des Clients verwenden.

```
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
]);
```

Allerdings ist die beste Möglichkeit, temporäre Anmeldeinformationen bereitzustellen, die Verwendung der `createCredentials()` Hilfsmethode aus dem `StsClient`. Diese Methode extrahiert die Daten aus einem AWS STS Ergebnis und erstellt das `Credentials` Objekt für Sie.

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

$s3Client = new S3Client([
```

```
'version'      => '2006-03-01',  
'region'       => 'us-west-2',  
'credentials' => $credentials  
]);
```

Weitere Informationen darüber, warum Sie möglicherweise temporäre Anmeldeinformationen in Ihrer Anwendung oder Ihrem Projekt verwenden müssen, finden Sie in der AWS STS Dokumentation unter [Szenarien für die Gewährung temporären Zugriffs](#).

Anonyme Clients im SDK for PHP erstellen

In einigen Fällen möchten Sie möglicherweise einen Client erstellen, der nicht mit Anmeldeinformationen verknüpft ist. Auf diese Weise können Sie anonymen Anforderungen an einen Service stellen.

Sie können beispielsweise sowohl Amazon S3-Objekte als auch CloudSearch Amazon-Domains so konfigurieren, dass anonymen Zugriff möglich ist.

Um einen anonymen Client zu erstellen, setzen Sie die 'credentials'-Option auf 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',  
]);
```

Verwenden Sie die AWS Common Runtime (AWS CRT) - Erweiterung im SDK for PHP

Die [AWS CRT-Bibliotheken](#) bieten grundlegende Funktionen mit guter Leistung und minimalem Platzbedarf für mehrere. AWS SDKs In diesem Thema wird erläutert, wann das AWS CRT vom SDK for PHP verwendet wird und wie die AWS CRT-Erweiterung installiert wird.

Wann muss die AWS CRT-Erweiterung installiert sein

Das SDK for PHP verwendet die Autorisierungs- und Prüfsummenfunktionen der AWS CRT-Bibliotheken. Die AWS CRT-Erweiterung ist erforderlich, wenn Sie arbeiten mit:

- [Multiregionale Amazon-S3-Zugriffspunkte](#)
- [EventBridge Globale Amazon-Endpunkte](#)
- [Ein CRC-32C-Prüfsummenalgorithmus in Amazon Simple Storage Service \(Amazon S3\)](#)

Wenn Sie eine der oben aufgeführten Funktionen verwenden und die AWS CRT-Erweiterung nicht in Ihrer PHP-Umgebung installiert ist, meldet das SDK for PHP eine Fehlermeldung und erinnert Sie daran, die Erweiterung zu installieren.

Installieren Sie die AWS Common Runtime (AWS CRT) -Erweiterung

Anweisungen zur Installation der AWS CRT-Erweiterung finden Sie auf der Hauptseite des [GitHubRepositorys für aws-crt-php](#)

Mit der AWS SDK für PHP Version 3

Dieses Kapitel enthält umfassende Anleitungen zur Verwendung der AWS SDK für PHP Version 3 für eine AWS-Services effektive Interaktion mit. Sie lernen grundlegende Techniken kennen, um Serviceanfragen zu stellen, Fehler zu behandeln und erweiterte SDK-Funktionen zu nutzen, um robuste PHP-Anwendungen zu AWS erstellen.

Ganz gleich, ob Sie eine einfache Anwendung oder ein komplexes System erstellen, die in diesem Kapitel beschriebenen Techniken helfen Ihnen dabei, Ihren Code zu optimieren, die Fehlerbehandlung zu verbessern und effiziente Muster für die Arbeit mit dem AWS-Services PHP-SDK zu implementieren.

Themen

- [AWS-Service Anfragen mit der AWS SDK für PHP Version 3 stellen](#)
- [Asynchrone Programmierung mit der AWS SDK für PHP Version 3](#)
- [Behandlung von Fehlern in der AWS SDK für PHP Version 3](#)
- [Handler und Middleware in der Version 3 AWS SDK für PHP](#)
- [Streams in der AWS SDK für PHP Version 3](#)
- [Verwendung von paginierten Ergebnissen in der AWS SDK für PHP Version 3](#)
- [Kellner in der AWS SDK für PHP Version 3](#)
- [JMESPath Ausdrücke in der AWS SDK für PHP Version 3](#)

AWS-Service Anfragen mit der AWS SDK für PHP Version 3 stellen

Überblick über den Workflow für SDK-Anfragen

Die Arbeit mit AWS SDK für PHP Version 3 folgt in allen Bereichen einem einheitlichen Muster AWS-Services. Der grundlegende Arbeitsablauf umfasst drei Hauptschritte:

1. [Einen Service-Client erstellen](#) — Instanziiieren Sie ein Client-Objekt für den Client, den AWS-Service Sie verwenden möchten.
2. [Operationen ausführen](#) — Rufen Sie Methoden auf dem Client auf, die Operationen in der API des Dienstes entsprechen.
3. [Ergebnisse verarbeiten](#) — Arbeiten Sie mit dem Array-ähnlichen Ergebnisobjekt, das bei Erfolg zurückgegeben wird, oder behandeln Sie jede Ausnahme, die bei einem Fehler ausgelöst wird.

In den folgenden Abschnitten wird jeder dieser Schritte detailliert beschrieben, beginnend mit der Erstellung und Konfiguration von Service-Clients.

Einen Basis-Serviceclient erstellen

Sie können einen Client erstellen, indem Sie dem Konstruktor eines Clients ein assoziatives Array von Optionen übergeben.

Importe

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

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

Beispiel-Code

```
//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'.
```

Informationen zum optionalen Parameter „Version“ finden Sie im Thema [Konfigurationsoptionen](#).

Beachten Sie, dass wir nicht explizit Anmeldeinformationen für den Client angegeben haben. Das liegt daran, dass das SDK die [standardmäßige Anbieterkette für Anmeldeinformationen](#) verwendet, um nach Anmeldeinformationen zu suchen.

Alle allgemeinen Client-Konfigurationsoptionen werden unter ausführlich beschrieben. [Client-Konstruktor-Optionen für die AWS SDK für PHP Version 3](#) Die Anzahl der Optionen, die einem Client zur Verfügung gestellt werden, kann je nach Client, den Sie erstellen, variieren. Diese benutzerdefinierten Clientkonfigurationsoptionen sind in der [API-Dokumentation](#) für jeden Client beschrieben.

Das obige Beispiel zeigt zwar die grundlegende Client-Erstellung, Sie können Ihre Service-Clients jedoch an spezifische Anforderungen anpassen. Ausführlichere Informationen zur Konfiguration von Service-Clients mithilfe von Code finden Sie unter [Konfiguration von Service-Clients im Code für die AWS SDK für PHP Version 3](#). Informationen zur Konfiguration von Service-Clients mithilfe externer

Konfigurationsdateien oder Umgebungsvariablen finden Sie unter [Service-Clients für die AWS SDK für PHP Version 3 extern konfigurieren](#).

Senden von Anforderungen

Sie können Serviceanfragen stellen, indem Sie die gleichnamige Methode für ein Client-Objekt aufrufen. Um beispielsweise den Amazon S3 [PutObjectS3-Vorgang](#) auszuführen, rufen Sie die `Aws\S3\S3Client::putObject()` Methode auf.

Importe

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

use Aws\S3\S3Client;
```

Beispiel-Code

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

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

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

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

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

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

Operationen, die für einen Client verfügbar sind, und die Struktur der Eingabe und Ausgabe werden zur Laufzeit basierend auf einer Servicebeschreibungsdatei definiert. Wenn Sie beim Erstellen eines Clients keinen `version` Parameter (z. B. „2006-03-01“ oder „latest“) des Servicemodells angeben, verwendet der Client standardmäßig die neueste Version. Das SDK findet die entsprechende Konfigurationsdatei basierend auf der bereitgestellten Version.

Operationsmethoden wie `putObject()` akzeptieren alle ein einzelnes Argument, ein assoziatives Array, das die Parameter der Operation darstellt. Die Struktur dieses Arrays (und die Struktur des Ergebnisobjekts) wird für jede Operation in der API-Dokumentation des SDK definiert (siehe z. B. die API-Dokumentation für die [putObject-Operation](#)).

Optionen für den HTTP-Handler

Sie können auch genau festlegen, wie der zugrunde liegende HTTP-Handler die Anforderung ausführt, indem Sie den speziellen Parameter `@http` verwenden. Die Optionen, die Sie in den Parameter `@http` aufnehmen können, entsprechen denen, die Sie beim Initialisieren des Clients mit der Client-Option „[http](#)“ festlegen können.

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

Mit Ergebnisobjekten arbeiten

Ausführen einer erfolgreichen Operation gibt ein `Aws\Result` Objekt zurück. Anstatt die XML- oder JSON-Rohdaten eines Service zurückzugeben, konvertiert das SDK die Antwortdaten in eine assoziative Array-Struktur. Es normalisiert einige Aspekte der Daten auf der Grundlage seiner Kenntnisse des spezifischen Services und der zugrunde liegenden Antwortstruktur.

Sie können auf Daten aus dem `Aws\Result` Objekt wie auf ein assoziatives PHP-Array zugreifen.

Importe

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

Beispiel-Code

```
// 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();
```

Der Inhalt des Ergebnisobjekts hängt von der ausgeführten Operation und der Version eines Service ab. Die Ergebnisstruktur jeder API-Operation ist in den API-Dokumenten für jede Operation dokumentiert.

Das SDK ist in eine [DSL](#) integriert [JMESPath](#), die zum Suchen und Bearbeiten von JSON-Daten oder in unserem Fall in PHP-Arrays verwendet wird. Das Ergebnisobjekt enthält eine `search()`-Methode, mit der Sie deklarativer Daten aus dem Ergebnis extrahieren können.

Beispiel-Code

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

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

Befehlsobjekte in der AWS SDK für PHP Version 3

Die AWS SDK für PHP verwendet das [Befehlsmuster](#), um die Parameter und den Handler zu kapseln, die zu einem späteren Zeitpunkt für die Übertragung einer HTTP-Anfrage verwendet werden.

Implizite Verwendung von Befehlen

Wenn Sie eine Clientklasse untersuchen, können Sie sehen, dass die den API-Operationen entsprechenden Methoden tatsächlich nicht vorhanden sind. Sie werden mit der magischen Methode `__call()` umgesetzt. Diese Pseudo-Methoden sind eigentlich Verknüpfungen, die die Verwendung von Befehlsobjekten durch das SDK einkapseln.

In der Regel müssen Sie nicht direkt mit Befehlsobjekten interagieren. Wenn Sie Methoden wie `Aws\S3\S3Client::putObject()` aufrufen, erstellt das SDK basierend auf den angegebenen Parametern ein Objekt `Aws\CommandInterface`, führt den Befehl aus und gibt ein ausgefülltes Objekt `Aws\ResultInterface` zurück (oder löst eine Ausnahme aus, wenn ein Fehler auftritt). Ein ähnlicher Ablauf tritt auf, wenn eine der Async-Methoden eines Clients (z. B. `Aws\S3\S3Client::putObjectAsync()`) aufgerufen wird: Der Client erstellt einen Befehl basierend auf den bereitgestellten Parametern, serialisiert eine HTTP-Anforderung, initiiert die Anforderung und gibt ein Promise zurück.

Die folgenden Beispiele sind funktional äquivalent.

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

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

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

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

Befehlsparameter

Alle Befehle unterstützen einige spezielle Parameter, die nicht Teil der API eines Services sind, sondern das Verhalten des SDK steuern.

@http

Mit diesem Parameter können Sie genau festlegen, wie der zugrunde liegende HTTP-Handler die Anfrage ausführt. Die Optionen, die Sie in den Parameter `@http` aufnehmen können, entsprechen denen, die Sie beim Initialisieren des Clients mit der Client-Option „[http](#)“ festlegen können.

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

@retries

Wie auch die Client-Option „[Wiederholungen](#)“ steuert `@retries`, wie oft ein Befehl wiederholt werden kann, bevor er als fehlgeschlagen gilt. Setzen Sie es auf `0`, um die Versuche zu deaktivieren.

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

Note

Wenn Sie Neuversuche auf einem Client deaktiviert haben, können Sie sie nicht selektiv für einzelne Befehle aktivieren, die an diesen Client übergeben werden.

Befehlsobjekte erstellen

Sie können einen Befehl mit der `getCommand()`-Methode eines Clients erstellen. Diese führt nicht sofort eine HTTP-Anfrage aus oder überträgt sie, sondern sie wird nur ausgeführt, wenn sie an die `execute()`-Methode des Clients übergeben wird. Dies gibt Ihnen die Möglichkeit, das Befehlsobjekt vor der Ausführung des Befehls zu ändern.

```
$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);
```

Befehl **HandlerList**

Wenn ein Befehl von einem Client erstellt wird, erhält er einen Klon des `Aws\HandlerList`-Objekts des Clients. Der Befehl erhält einen Klon der Handlerliste des Clients, damit ein Befehl benutzerdefinierte Middleware und Handler verwenden kann, die andere vom Client ausgeführte Befehle nicht beeinflussen.

Dies bedeutet, dass Sie einen anderen HTTP-Client pro Befehl (z. B. `Aws\MockHandler`) verwenden und benutzerdefiniertes Verhalten pro Befehl über Middleware hinzufügen können. Im folgenden Beispiel wird eine `MockHandler` verwendet, um Scheinergebnisse anstelle von tatsächlichen HTTP-Anforderungen zu erstellen.

```
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'
```

Zusätzlich zum Ändern des vom Befehl verwendeten Handlers können Sie dem Befehl auch benutzerdefinierte Middleware hinzufügen. Das folgende Beispiel verwendet die Middleware `tap`, die als Beobachter in der Handlerliste fungiert.

```
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

Mit der `Aws\CommandPool` können Sie Befehle gleichzeitig mit einem Iterator ausführen, der `Aws\CommandInterface`-Objekte liefert. Die `CommandPool` stellt sicher, dass eine konstante Anzahl von Befehlen gleichzeitig ausgeführt wird, während über die Befehle im Pool iteriert wird (wenn Befehle abgeschlossen sind, werden mehr ausgeführt, um eine konstante Poolgröße sicherzustellen).

Hier ist ein sehr einfaches Beispiel, um nur einige Befehle mit einer `CommandPool` zu senden.

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

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

$bucket = 'amzn-s3-demo-bucket';
```

```
$commands = [  
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'a']),  
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'b']),  
    $client->getCommand('HeadObject', ['Bucket' => $bucket, 'Key' => 'c'])  
];  
  
$pool = new CommandPool($client, $commands);  
  
// Initiate the pool transfers  
$promise = $pool->promise();  
  
// Force the pool to complete synchronously  
$promise->wait();
```

Dieses Beispiel ist für die `CommandPool` ziemlich unterentwickelt. Versuchen wir es mit einem komplexeren Beispiel. Nehmen wir an, Sie möchten Dateien auf der Festplatte in einen Amazon S3 S3-Bucket hochladen. Um eine Liste der Dateien von der Festplatte zu erhalten, können wir `DirectoryIterator` von PHP verwenden. Dieser Iterator liefert `SplFileInfo` Objekte. Der `CommandPool` akzeptiert einen Iterator, der `Aws\CommandInterface`-Objekte liefert, also bilden wir die `SplFileInfo`-Objekte ab, um `Aws\CommandInterface`-Objekte zurückzugeben.

```
<?php  
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\S3\S3Client;  
use Aws\CommandPool;  
use Aws\CommandInterface;  
use Aws\ResultInterface;  
use GuzzleHttp\Promise\PromiseInterface;  
  
// Create the client  
$client = new S3Client([  
    'region' => 'us-standard',  
    'version' => '2006-03-01'  
]);  
  
$fromDir = '/path/to/dir';  
$toBucket = 'amzn-s3-demo-bucket';  
  
// Create an iterator that yields files from a directory  
$files = new DirectoryIterator($fromDir);
```

```

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

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

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

```

```

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

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

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

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

```

CommandPool-Konfiguration

Der `Aws\CommandPool` Konstruktor akzeptiert verschiedene Konfigurationsoptionen.

Nebenläufigkeit (aufrufbar|int)

Maximale Anzahl von Befehlen, die gleichzeitig ausgeführt werden. Stellen Sie eine Funktion bereit, um den Pool dynamisch zu skalieren. Der Funktion wird die aktuelle Anzahl der ausstehenden Anforderungen bereitgestellt, und es wird erwartet, dass sie eine Ganzzahl zurückgibt, die die neue Größe des Pools darstellt.

vor der Aktualisierung (abrufbar)

Funktion, die vor dem Senden jedes Befehls aufgerufen wird. Die `before` Funktion akzeptiert den Befehl und den Schlüssel des Iterators des Befehls. Sie können den Befehl nach Bedarf in der Funktion `before` mutieren, bevor Sie den Befehl senden.

erfüllt (aufrufbar)

Funktion, die aufgerufen wird, wenn ein Promise erfüllt ist. Die Funktion enthält das Ergebnisobjekt, die ID des Iterators, von dem das Ergebnis stammt, und das zusammengefasste Promise, das aufgelöst oder zurückgewiesen werden kann, wenn der Pool kurzgeschlossen werden muss.

abgelehnt (Callable)

Funktion, die aufgerufen wird, wenn ein Promise abgelehnt wird. Der Funktion wird ein `Aws\Exception` Objekt zur Verfügung gestellt, die ID des Iterators, von der die Ausnahme kam, und

das zusammengefasste Promise, das aufgelöst oder zurückgewiesen werden kann, wenn der Pool kurzgeschlossen werden muss.

Manuelle Speicherbereinigung zwischen Befehlen

Wenn Sie bei großen Befehlspools das Speicherlimit ausreizen, ist die unter Umstände auf zyklische Referenzen zurückzuführen, die vom SDK konfiguriert wurden und noch nicht von der [PHP-Speicherbereinigung](#) gelöscht wurden, als das Speicherlimit erreicht wurde. Manuelles Aufrufen des Bereinigungsalgorithmus zwischen Befehlen ermöglicht unter Umständen die Löschung der Zyklen, bevor das Limit erreicht wird. Das folgende Beispiel erstellt einen `CommandPool`, der den Bereinigungsalgorithmus mithilfe eines Callbacks aufruft, bevor er die einzelnen Befehle versendet. Beachten Sie, dass das Aufrufen der Speicherbereinigung zulasten der Leistung geht und die optimale Nutzung von Ihrem Anwendungsfall und Ihrer Umgebung abhängt.

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

Asynchrone Programmierung mit der AWS SDK für PHP Version 3

Sie können Befehle gleichzeitig mit den asynchronen Funktionen des SDKs senden. Sie können Anfragen asynchron senden, indem Sie einen Operationsnamen mit `Async` suffizieren. Dies initiiert die Anfrage und gibt ein Promise zurück.

Das Promise wird mit dem Ergebnisobjekt bei Erfolg erfüllt oder mit einer Ausnahme bei einem Fehler abgelehnt. Auf diese Weise können Sie mehrere Promises erstellen und veranlassen, dass HTTP-Anforderungen gleichzeitig gesendet werden, wenn der zugrunde liegende HTTP-Handler die Anforderungen überträgt.

Importe

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

Beispiel-Code

```
// 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();
```

Sie können die Versprechung eines Promises erzwingen, indem Sie die Methode `wait` des Promises synchron ausführen. Das Erzwingen des Promises zum Vervollständigen „packt“ auch standardmäßig den Status des Promises „aus“, was bedeutet, dass es entweder das Ergebnis des Promises zurückgibt oder die aufgetretene Ausnahme auslöst. Beim Aufruf von `wait()` bei einem Promise blockiert der Prozess, bis die HTTP-Anfrage abgeschlossen ist und das Ergebnis gefüllt ist oder eine Ausnahme ausgelöst wird.

Wenn Sie das SDK mit einer Ereignisschleifenbibliothek verwenden, blockieren Sie keine Ergebnisse. Verwenden Sie stattdessen die Methode `then()` eines Ergebnisses, um auf eine Zusage zuzugreifen, die nach Abschluss der Operation aufgelöst oder zurückgewiesen wird.

Importe

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

Beispiel-Code

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

Versprechen in der AWS SDK für PHP Version 3

Die AWS SDK für PHP Verwendung verspricht, asynchrone Workflows zu ermöglichen, und diese Asynchronität ermöglicht das gleichzeitige Senden von HTTP-Anfragen. Die vom SDK verwendete Promise-Spezifikation lautet [Promises/A+](#).

Was ist ein Versprechen?

Ein Promise repräsentiert das Ergebnis einer asynchronen Operation. Der primäre Weg der Interaktion mit einem Promise ist durch seine `then`-Methode. Diese Methode registriert Callbacks, um entweder den möglichen Wert eines Promises oder den Grund zu erhalten, warum das Promise nicht erfüllt werden kann.

The AWS SDK für PHP stützt sich bei der Implementierung seiner Promises auf das Composer-Paket [guzzlehttp/promises](#). Guzzle unterstützt blockierende und nicht blockierende Workflows und kann mit jeder nicht blockierenden Ereignisschleife verwendet werden.

Note

HTTP-Anfragen werden gleichzeitig AWS SDK für PHP unter Verwendung eines einzigen Threads gesendet, wobei nicht blockierende Aufrufe verwendet werden, um eine oder mehrere HTTP-Anfragen zu übertragen und gleichzeitig auf Statusänderungen zu reagieren (z. B. Versprechen zu erfüllen oder abzulehnen).

Promises im SDK

Promises werden im gesamten SDK verwendet. [Promises werden beispielsweise in den meisten vom SDK bereitgestellten allgemeinen Abstraktionen verwendet: Paginatoren, Kellner, Befehlspools, mehrteilige Uploads, S3-Übertragungen usw. `directory/bucket`](#)

Alle Clients, die das SDK bereitstellt, geben Rück-Promises, wenn Sie eine der Suffixmethoden `Async` aufrufen. Der folgende Code zeigt beispielsweise, wie ein Versprechen zum Abrufen der Ergebnisse eines Amazon DynamoDB `DescribeTable`-Vorgangs erstellt wird.

```
$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']);
```

Beachten Sie, dass Sie entweder `describeTable` oder `describeTableAsync` aufrufen können. Diese Methoden sind magische `__call` Methoden auf einem Client, die durch das API-Modell und version Nummer mit dem Client verbunden sind. Indem Methoden wie `describeTable` ohne das Suffix `Async` aufgerufen werden, blockiert der Client, während er eine HTTP-Anfrage sendet und entweder ein `Aws\ResultInterface`-Objekt zurückgibt oder eine `Aws\Exception\AwsException` auslöst. Durch Suffix des Operationsnamens mit `Async` (d. h. `describeTableAsync`) erstellt der Client ein Promise, das schließlich mit einem Objekt `Aws\ResultInterface` erfüllt oder mit einer `Aws\Exception\AwsException` abgelehnt wird.

Important

Wenn das Promise zurückgegeben wird, ist das Ergebnis möglicherweise bereits angekommen (z. B. bei Verwendung eines Mock-Handlers) oder die HTTP-Anforderung wurde möglicherweise nicht initiiert.

Sie können einen Callback mit dem Promise mit der Methode `then` registrieren. Diese Methode akzeptiert zwei Callbacks, `$onFulfilled` und `$onRejected`, die beide optional sind. Der Callback `$onFulfilled` wird aufgerufen, wenn das Promise erfüllt ist, und der Callback `$onRejected` wird aufgerufen, wenn das Promise abgelehnt wird (d.h. es ist fehlgeschlagen).

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

Gleichzeitige Ausführung von Befehlen

Mehrere Promises können zusammen so zusammengesetzt werden, dass sie gleichzeitig ausgeführt werden. Dies kann erreicht werden, indem das SDK in eine nicht blockierende Ereignisschleife integriert wird oder indem mehrere Promises aufgebaut und darauf gewartet wird, dass sie gleichzeitig ausgeführt werden.

```
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

Das [CommandPool](#) bietet einen leistungsfähigeren Mechanismus für die gleichzeitige Ausführung mehrerer API-Operationen.

Versprechen verketteten

Einer der besten Aspekte von Promises ist, dass sie zusammensetzbar sind, sodass Sie Transformations-Pipelines erstellen können. Promise werden durch Verkettung then von Callbacks mit nachfolgenden then Callbacks zusammengestellt. Der Rückgabewert einer then-Methode ist ein Promise, das basierend auf dem Ergebnis der bereitgestellten Callbacks erfüllt oder abgelehnt wird.

```
$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

Der Rückgabewert eines Zusage-Callbacks ist das Argument `$value`, das an Downstream-Promises übergeben wird. Wenn Sie Downstream-Promise-Ketten einen Wert bereitstellen möchten, müssen Sie einen Wert in der Callback-Funktion zurückgeben.

Ablehnung, Weiterleitung

Sie können einen Callback registrieren, der aufgerufen werden soll, wenn ein Promise abgelehnt wird. Wenn in einem Callback eine Ausnahme ausgelöst wird, wird das Promise mit der Ausnahme abgelehnt und die nächsten Promise in der Kette werden mit der Ausnahme abgelehnt. Wenn Sie einen Wert aus einem `$onRejected`-Callback erfolgreich zurückgeben, werden die nächsten Promises in der Promise-Kette mit dem Rückgabewert aus dem Callback `$onRejected` erfüllt.

Ich warte auf Versprechen

Sie können die Promises synchron erzwingen, indem Sie die `wait` -Methode eines Promises verwenden.

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

Wenn beim Aufrufen der Funktion `wait` eines Promises eine Ausnahme auftritt, wird das Promise mit der Ausnahme abgelehnt und die Ausnahme wird ausgelöst.

```
use Aws\Exception\AwsException;  
  
$promise = $client->listTablesAsync();  
  
try {  
    $result = $promise->wait();  
} catch (AwsException $e) {  
    // Handle the error  
}
```

Der Aufruf von `wait` auf einem erfüllten Promise löst die Wartefunktion nicht aus. Es gibt einfach den zuvor gelieferten Wert zurück.

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

Der Aufruf von `wait` bei einem abgelehnten Promise löst eine Ausnahme aus. Wenn der Ablehnungsgrund eine Instance von `\Exception` ist, wird der Grund geworfen. Andernfalls wird eine `GuzzleHttp\Promise\RejectionException` ausgelöst und der Grund kann durch Aufrufen der `getReason`-Methode der Ausnahme erhalten werden.

Note

API-Operationsaufrufe in AWS SDK für PHP werden mit Unterklassen der `Aws\Exception\AwsException` Klasse zurückgewiesen. Es ist jedoch möglich, dass der Grund, der an eine `then`-Methode geliefert wird, ein anderer ist, weil eine benutzerdefinierte Middleware hinzugefügt wird, die einen Ablehnungsgrund ändert.

Versprechen stornieren

Promise können mit der `cancel()`-Methode eines Promises abgebrochen werden. Wenn eine Zusage bereits gelöst wurde, hat der Aufruf von `cancel()` keine Wirkung. Das Abbrechen eines Promises löscht das Promise und alle Promises, die auf die Lieferung vom Promise warten. Ein gelöscht Promise wird mit einer `GuzzleHttp\Promise\RejectionException` abgelehnt.

Versprechen kombinieren

Sie können Promise zu zusammengefassten Promises kombinieren, um anspruchsvollere Workflows zu erstellen. Das Paket `guzzlehttp/promise` enthält verschiedene Funktionen, mit denen Sie Promises kombinieren können.

Die API-Dokumentation für alle Funktionen zur Sammlung von Versprechen finden Sie unter [namespace- GuzzleHttp .Promise](#).

`each` und `each_limit`

Verwenden Sie die [CommandPool](#) Option, wenn Sie eine Aufgabenwarteschlange mit `Aws\CommandInterface` Befehlen haben, die gleichzeitig mit einer festen Poolgröße ausgeführt werden sollen (die Befehle können sich im Speicher befinden oder von einem Lazy-Iterator abgerufen werden). Die `CommandPool` stellt sicher, dass eine feste Anzahl von Befehlen gleichzeitig gesendet wird, bis der bereitgestellte Iterator erschöpft ist.

Das `CommandPool` funktioniert nur mit den gleichen Client-Befehlen, die ausgeführt werden. Sie können die Funktion `GuzzleHttp\Promise\each_limit` verwenden, um Sendebefehle verschiedener Clients gleichzeitig mit einer festen Poolgröße auszuführen.

```
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();
```

Versprich Coroutinen

Eine der leistungstärkeren Funktionen der Guzzle Promise-Bibliothek ist, dass Sie Promise-Coroutinen verwenden können, die das Schreiben von asynchronen Arbeitsabläufen mehr wie das Schreiben von traditionellen synchronen Arbeitsabläufen erscheinen lassen. Tatsächlich verwendet das AWS SDK für PHP Coroutine-Promises in den meisten Abstraktionen auf hoher Ebene.

Stellen Sie sich vor, Sie möchten mehrere Buckets erstellen und eine Datei in den Bucket hochladen, sobald der Bucket verfügbar wird. Sie möchten dies alles gleichzeitig tun, damit es so schnell wie möglich geschieht. Sie können dies leicht tun, indem Sie mehrere Coroutine-Promises mit der Funktion `all()` kombinieren.

```
use GuzzleHttp\Promise;

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

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

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

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

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

Behandlung von Fehlern in der AWS SDK für PHP Version 3

Synchrone Fehlerbehandlung

Wenn beim Ausführen einer Operation ein Fehler auftritt, wird eine Ausnahme ausgelöst. Aus diesem Grund verwenden Sie zur Behandlung von Fehlern in Ihrem Code try/catch-Blöcke um Ihre Operationen herum. Das SDK löst servicespezifische Ausnahmen aus, wenn ein Fehler auftritt.

Das folgende Beispiel verwendet die `Aws\S3\S3Client`. Wenn ein Fehler vorliegt, wird die ausgelöste Ausnahme vom Typ `Aws\S3\Exception\S3Exception` sein. Alle servicespezifischen Ausnahmen, die das SDK auslöst, erstrecken sich von der Klasse `Aws\Exception` `\AwsException`. Diese Klasse enthält nützliche Informationen zum Fehler einschließlich der Anforderungs-ID, des Fehlercodes und des Fehlertyps. Für einige Services, die diese Klasse unterstützen, werden Antwortdaten in eine assoziative Array-Struktur (ähnelt `Aws\Result`-Objekten) umgewandelt, auf die wie auf normale assoziative PHP-Arrays zugegriffen werden kann. Die Methode `toArray()` gibt Daten dieser Art zurück (sofern vorhanden).

Importe

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

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

Beispiel-Code

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

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

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

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

Asynchrone Fehlerbehandlung

Ausnahmen werden nicht ausgelöst, wenn das Senden von asynchronen Anforderungen. Stattdessen müssen Sie die Methode `then()` oder `otherwise()` des zurückgegebenen Promise verwenden, um das Ergebnis oder den Fehler zu erhalten.

Importe

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

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

Beispiel-Code

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

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

Sie können das Promise „auspacken“ und stattdessen die Ausnahme auslösen.

Importe

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

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

Beispiel-Code

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

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

Handler und Middleware in der Version 3 AWS SDK für PHP

Der Hauptmechanismus für die Erweiterung von AWS SDK für PHP ist der Einsatz von Handlern und Middleware. Jede SDK-Client-Klasse besitzt eine `Aws\HandlerList`-Instance, die über die

`getHandlerList()`-Methode eines Clients erreichbar ist. Sie können die `HandlerList` eines Clients abrufen und ändern, um Client-Verhaltensweisen hinzuzufügen oder zu entfernen.

Handler

Ein Handler ist eine Funktion, die die eigentliche Transformation eines Befehls und einer Anfrage in ein Ergebnis durchführt. Ein Handler sendet typischerweise HTTP-Anfragen. Handler können mit Middleware konstruiert werden, um ihr Verhalten zu verbessern. Ein Handler ist eine Funktion, die eine `Aws\CommandInterface` und eine `Psr\Http\Message\RequestInterface` akzeptiert und ein `Promise` zurückgibt, das mit einer `Aws\ResultInterface` erfüllt oder mit einem `Aws\Exception\AwsException` Grund abgelehnt wird.

Hier ist ein Handler, der für jeden Aufruf das gleiche modellhafte Ergebnis liefert.

```
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);
};
```

Sie können diesen Handler mit einem SDK-Client verwenden, indem Sie eine `handler`-Option im Konstruktor eines Clients angeben.

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

Sie können den Handler eines Clients auch ändern, nachdem er erstellt wurde. Dazu verwenden Sie die `setHandler`-Methode eines `Aws\ClientInterface`.

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

 Note

Um den Handler eines Clients mit mehreren Regionen nach seiner Erstellung zu ändern, verwenden Sie die `useCustomHandler` Methode eines `Aws\MultiRegionClient`

```
$multiRegionClient->useCustomHandler($myHandler);
```

Handler simulieren

Wir empfehlen die Verwendung von `MockHandler` bei der Erstellung von Tests, die das SDK verwenden. Du kannst den `Aws\MockHandler` benutzen, um modellhafte Ergebnisse zurückzugeben oder modellhafte Ausnahmen aufzuwerfen. Sie stellen Ergebnisse oder Ausnahmen in die Warteschlange und stellen sie dann in der `MockHandler` FIFO-Reihenfolge aus der Warteschlange.

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

$mock = new MockHandler();

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

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

// Create a client with the mock handler
$client = new DynamoDbClient([
    'region' => 'us-west-2',
    'version' => 'latest',
    'handler' => $mock
]);

// Result object response will contain ['foo' => 'bar']
```

```
$result = $client->listTables();

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

Middleware

Middleware ist eine spezielle Art von High-Level-Funktion, die das Verhalten bei der Übertragung eines Befehls erweitert und an einen „nächsten“ Handler delegiert. Middleware-Funktionen akzeptieren eine `Aws\CommandInterface` und eine `Psr\Http\Message\RequestInterface` und geben ein `Promise` zurück, das mit einer `Aws\ResultInterface` erfüllt oder mit einem `Aws\Exception\AwsException`-Grund abgelehnt wird.

Eine Middleware ist eine Funktion höherer Ordnung, die einen Befehl, eine Anforderung oder ein Ergebnis beim Durchlaufen der Middleware ändert. Eine Middleware hat die folgende Form.

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

Eine Middleware erhält einen Befehl zum Ausführen und ein optionales Anfrage-Objekt. Die Middleware kann die Anfrage und den Befehl erweitern oder sie unverändert lassen. Eine Middleware ruft dann das nächste Handle in der Kette auf oder kann den nächsten Handler kurzschließen und ein `Promise` zurückgeben. Das `Promise`, das durch den Aufruf des nächsten Handlers erzeugt wird, kann dann mit der Methode `then` des `Promise` erweitert werden, um das eventuelle Ergebnis oder den Fehler zu ändern, bevor das `Promise` zurück auf den Stack der Middleware gesendet wird.

HandlerList

Das SDK verwendet eine `Aws\HandlerList` zum Verwalten der Middleware und der Handler, wenn es einen Befehl ausführt. Jeder SDK-Client besitzt eine `HandlerList`, und diese `HandlerList` wird geklont und jedem Befehl hinzugefügt, den ein Client erstellt. Sie können eine Middleware und einen Standard-Handler für jeden von einem Client erstellten Befehl anfügen, indem Sie eine Middleware zur `HandlerList` des Clients hinzufügen. Sie können Middleware zu bestimmten Befehlen hinzufügen oder entfernen, indem Sie die `HandlerList` für einen bestimmten Befehl ändern.

Eine `HandlerList` stellt einen Stack von Middleware dar, der verwendet wird, um einen Handler zu kapseln. Für die Verwaltung der Liste der Middleware und die Reihenfolge, in der sie einen Handler kapseln, unterteilt die `HandlerList` den Middleware-Stack in benannte Schritte, die Teil des Lebenszyklus der Übertragung eines Befehls sind:

1. `init` – Standardparameter hinzufügen
2. `validate` – Validieren der erforderlichen Parameter
3. `build` – Serialisieren einer HTTP-Anforderung für das Senden
4. `sign` – Signieren der serialisierten HTTP-Anforderung
5. `<handler>` (kein Schritt, aber führt die tatsächliche Übertragung durch)

`init`

Dieser Lifecycle-Schritt stellt die Initialisierung eines Befehls dar, und eine Anforderung wurde noch nicht serialisiert. Dieser Schritt wird in der Regel verwendet, um einem Befehl Standardparameter hinzuzufügen.

Sie können dem `init`-Schritt Middleware unter Verwendung der `appendInit`- und `prependInit`-Methoden hinzufügen, wobei `appendInit` die Middleware am Ende der `prepend`-Liste hinzufügt, während `prependInit` die Middleware am Anfang der `prepend`-Liste hinzufügt.

```
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');
```

validieren

Dieser Lebenszyklus-Schritt dient der Validierung der Eingabeparameter eines Befehls.

Sie können dem `validate`-Schritt Middleware unter Verwendung der `appendValidate`- und `prependValidate`-Methoden hinzufügen, wobei `appendValidate` die Middleware am Ende der `validate`-Liste hinzufügt, während `prependValidate` die Middleware am Anfang der `validate`-Liste hinzufügt.

```
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

Dieser Lebenszyklus-Schritt wird verwendet, um eine HTTP-Anforderung für den auszuführenden Befehl zu serialisieren. Nachgelagerte Lebenszyklus-Ereignisse erhalten einen Befehl und eine PSR-7 HTTP-Anforderung.

Sie können dem `build`-Schritt Middleware unter Verwendung der `appendBuild`- und `prependBuild`-Methoden hinzufügen, wobei `appendBuild` die Middleware am Ende der `build`-Liste hinzufügt, während `prependBuild` die Middleware am Anfang der `build`-Liste hinzufügt.

```
use Aws\Middleware;

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

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

sign

Dieser Lebenszyklus-Schritt wird normalerweise verwendet, um eine HTTP-Anforderung zu signieren, bevor sie gesendet wird. Um Signaturfehler zu vermeiden, sollten Sie davon absehen, eine HTTP-Anforderung zu ändern, nachdem sie signiert wurde.

Dies ist der letzte Schritt in der `HandlerList`, bevor die HTTP-Anforderung durch einen Handler übertragen wird.

Sie können dem `sign`-Schritt Middleware unter Verwendung der `appendSign`- und `prependSign`-Methoden hinzufügen, wobei `appendSign` die Middleware am Ende der `sign`-Liste hinzufügt, während `prependSign` die Middleware am Anfang der `sign`-Liste hinzufügt.

```
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');
```

Verfügbare Middleware

Das SDK bietet verschiedene Middleware, den Sie verwenden können, um das Verhalten eines Clients oder die Ausführung eines Befehls zu erweitern.

mapCommand

Die `Aws\Middleware::mapCommand` Middleware ist nützlich, wenn Sie einen Befehl ändern müssen, bevor der Befehl als HTTP-Anforderung serialisiert wird. Beispielsweise kann `mapCommand` verwendet werden, um eine Validierung durchzuführen oder Standardparameter hinzuzufügen. Die `mapCommand`-Funktion akzeptiert eine aufrufbare Funktion, die ein `Aws\CommandInterface`-Objekt entgegennimmt und ein `Aws\CommandInterface`-Objekt zurückgibt.

```

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

Die `Aws\Middleware::mapRequest` Middleware ist nützlich, wenn Sie einen Befehl ändern müssen, nachdem er serialisiert wurde, aber bevor er gesendet wird. Beispielsweise kann sie verwendet werden, um einer Anforderung benutzerdefinierte HTTP-Header hinzuzufügen. Die `mapRequest`-Funktion akzeptiert eine aufrufbare Funktion, die ein `Psr\Http\Message\RequestInterface`-Argument entgegennimmt und ein `Psr\Http\Message\RequestInterface`-Objekt zurückgibt.

```

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'
);

```

```
);
```

Wenn der Befehl ausgeführt wird, wird er mit dem benutzerdefinierten Header gesendet.

Important

Beachten Sie, dass die Middleware der Handler-Liste am Ende des `build`-Schritts hinzugefügt wurde. Damit soll sichergestellt werden, dass eine Anforderung erstellt wurde, bevor diese Middleware aufgerufen wird.

mapResult

Die `Aws\Middleware::mapResult` Middleware ist nützlich, wenn Sie das Ergebnis einer Befehlsausführung ändern müssen. Die `mapResult`-Funktion akzeptiert eine aufrufbare Funktion, die ein `Aws\ResultInterface`-Argument entgegennimmt und ein `Aws\ResultInterface`-Objekt zurückgibt.

```
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;
    })
);
```

Wenn der Befehl ausgeführt wird, enthält das zurückgegebene Ergebnis ein `foo`-Attribut.

history

Die `history` Middleware ist nützlich zum Testen, ob das SDK die Befehle ausgeführt hat, wie erwartet, ob es die HTTP-Anforderungen gesendet hat, wie erwartet, und ob die Ergebnisse erzeugt

wurden, wie erwartet. Es ist im Wesentlichen eine Middleware, die sich ähnlich verhält wie der Verlauf eines Web-Browsers.

```
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));
```

Ein `Aws\History`-Verlaufscontainer speichert standardmäßig 10 Einträge, bevor er Einträge löscht. Sie können die Anzahl der Einträge anpassen, indem Sie die Anzahl der beizubehaltenden Einträge an den Konstruktor übergeben.

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

Sie können den Verlaufscontainer nach der Ausführung von Anforderungen, die die Verlaufs-Middleware durchlaufen, überprüfen.

```
// 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

Die tap Middleware wird als Beobachter verwendet. Sie können diese Middleware verwenden, um Funktionen aufzurufen, wenn Sie Befehle durch die Middleware-Kette senden. Die tap Funktion akzeptiert eine aufrufbare Funktion, die die Aws\CommandInterface entgegennimmt, ebenso wie eine optionale Psr\Http\Message\RequestInterface, die gerade ausgeführt wird.

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

Benutzerdefinierte Handler erstellen

Ein Handler ist einfach eine Funktion, die ein `Aws\CommandInterface`-Objekt und ein `Psr\Http\Message\RequestInterface`-Objekt entgegennimmt und eine `GuzzleHttp\Promise\PromiseInterface` zurückgibt, die mit einer `Aws\ResultInterface` erfüllt oder mit einer `Aws\Exception\AwsException` abgelehnt wird.

Obwohl das SDK hat mehrere `@http`-Optionen bietet, muss ein Handler nur wissen, wie die folgenden Optionen verwendet werden:

- [connect_timeout](#)
- [debug](#)
- [decode_content](#) (optional)
- [Verzögerung](#)
- [progress](#) (optional)
- [proxy](#)
- [sink](#)
- [synchronous](#) (optional)
- [stream](#) (optional)
- [timeout](#)
- [verify](#)
- `http_stats_receiver` (optional) – Eine Funktion zum Aufrufen mit einem assoziativen Array von HTTP-Übertragungsstatistiken, wenn Sie über den [stats](#)-Konfigurationsparameter angefordert wurden.

Wenn die Option nicht als optional angegeben ist, MUSS ein Handler in der Lage sein, die Option zu verarbeiten, oder er MUSS ein abgelehntes Promise zurückgeben.

Zusätzlich zur Behandlung bestimmter `@http` Optionen MUSS ein Handler einen User-Agent Header hinzufügen, der die folgende Form hat, wobei „3.X“ durch ersetzt werden kann `Aws\Sdk::VERSION` und „HandlerSpecificData/version...“ durch Ihre handlerspezifische User-Agent-Zeichenfolge ersetzt werden sollte.

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

Streams in der AWS SDK für PHP Version 3

[Im Rahmen der Integration des PSR-7-HTTP-Nachrichtenstandards](#) AWS SDK für PHP verwendet der PSR-7 `StreamInterface` intern als Abstraktion gegenüber PHP-Streams. Jeder Befehl, dessen Eingabefeld als Blob definiert ist, wie z. B. der Body Parameter eines [S3::PutObject -Befehls](#), kann mit einer Zeichenfolge, einer PHP-Stream-Ressource oder einer Instanz von `Psr\Http\Message\StreamInterface` erfüllt werden.

Warning

Das SDK übernimmt alle PHP-Stream-Rohressourcen, die als Eingabeparameter für einen Befehl übergeben werden. Der Stream wird in Ihrem Namen verbraucht und geschlossen. Wenn Sie einen Stream zwischen einer SDK-Operation und Ihrem Code teilen müssen, schließen Sie ihn in eine Instance von `GuzzleHttp\Psr7\Stream` ein, bevor Sie ihn als Befehlsparameter aufnehmen. Das SDK verwendet den Stream, sodass Ihr Code die Bewegung des internen Cursors des Streams berücksichtigen muss. Guzzle-Streams rufen `fclose` für die zugrunde liegende Stream-Ressource auf, wenn sie vom PHP-Garbage Collector vernichtet werden, so müssen Sie den Stream also nicht selbst schließen.

Stream-Dekorateur

Guzzle bietet mehrere Stream-Decoratoren, mit denen Sie steuern können, wie das SDK und Guzzle mit der Streaming-Ressource interagieren, die als Eingabeparameter für einen Befehl bereitgestellt wird. Diese Decoratoren können verändern, wie Handler in einem bestimmten Stream lesen und suchen können. Die folgende Liste ist unvollständig; weitere finden Sie im [GuzzleHttpPsr7-Repository](#).

AppendStream

[GuzzleHttp\Psr7\AppendStream](#)

Lesen aus mehreren Streams nacheinander.

```
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](#)

Wird verwendet, um die Suche nach zuvor gelesenen Bytes in nicht durchsuchbaren Streams zu ermöglichen. Dies kann nützlich sein, wenn die Übertragung eines nicht durchsuchbaren Entity-Rumpfs fehlschlägt, weil der Stream auf den Anfang zurückgesetzt werden muss (z. B. aufgrund einer Weiterleitung). Daten, die aus dem Remote-Stream gelesen werden, werden in einem temporären PHP-Stream gepuffert, sodass zuvor gelesene Bytes zuerst im Speicher und dann auf der Festplatte zwischengespeichert werden.

```
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](#)

Verwendet den `zlib.inflate`-Filter von PHP zum Erweitern oder Komprimieren von gezippten Inhalten.

Dieser Stream-Decorator überspringt die ersten 10 Bytes des vorgegebenen Streams, um den gzip-Header zu entfernen, wandelt den Stream in eine PHP-Stream-Ressource um und fügt ihn dann dem `zlib.inflate`-Filter hinzu. Der Stream wird dann wieder in eine Guzzle-Stream-Ressource umgewandelt, die als Guzzle-Stream verwendet werden kann.

LazyOpenStream

[GuzzleHttp\Psr7\LazyOpenStream](#)

Liest eine Datei langsam, die erst nach einer I/O-Operation auf dem Stream geöffnet wird, bzw. schreibt in diese.

```
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](#)

Wird verwendet, um eine Teilmenge oder ein Segment eines vorhandenen Stream-Objekts lesen. Dies kann nützlich sein, um eine große Datei in kleinere Teile aufzuteilen, die dann in Blöcken gesendet werden (z. B. die Amazon S3 Multipart Upload API).

```
use GuzzleHttp\Psr7;

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

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

NoSeekStream

[GuzzleHttp\Psr7\NoSeekStream](#)

Verpackt einen Stream und lässt keine Suche zu.

```
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](#)

Stellt einen schreibgeschützten Datenstream bereit, der Daten aus einer aufrufbaren PHP-Funktion pumpt.

Beim Aufrufen des bereitgestellten Callables PumpStream übergibt der die Menge der zum Lesen angeforderten Daten an das Callable. Der aufrufbare Funktion kann diesen Wert ignorieren und weniger oder mehr Bytes als gewünscht zurückgeben. Alle zusätzlichen Daten, die vom bereitgestellten Callable zurückgegeben werden, werden intern gepuffert, bis sie mit der Funktion `read()` von gelöscht werden. PumpStream Die bereitgestellte aufrufbare Funktion MUSS `false` zurückgeben, wenn es keine weiteren Daten zu lesen gibt.

Implementierung von Stream-Dekoratoren

Das Erstellen eines Stream-Decorators ist dank [GuzzleHttp\Psr7](#) sehr einfach.

StreamDecoratorTrait Dieses Trait bietet Methoden, die `Psr\Http\Message\StreamInterface` implementieren, indem sie sich als Proxy für einen zugrundeliegenden Stream verhalten. Sie verwenden den `use` einfach mit `StreamDecoratorTrait` und implementieren Ihre benutzerdefinierten Methoden.

Angenommen, wir wollten jedes Mal eine bestimmte Funktion aufrufen, wenn das letzte Byte aus einem Stream gelesen wurde. Dies könnte durch Überschreiben der `read()`-Methode implementiert werden.

```
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;
    }
}
```

Dieser Decorator könnte in jedem vorhandenen Stream hinzugefügt und wie dieser verwendet werden.

```
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!"
```

Verwendung von paginierten Ergebnissen in der AWS SDK für PHP

Version 3

Einige AWS Serviceoperationen sind paginiert und antworten mit verkürzten Ergebnissen. Beispielsweise gibt der Amazon S3 `ListObjects` S3-Vorgang nur bis zu 1.000 Objekte gleichzeitig zurück. Operationen wie diese (häufig mit dem Präfix „list“ oder „describe“) erfordern, dass nachfolgende Anforderungen mit Token- (oder Markierungs-) Parametern durchgeführt werden, um den gesamten Satz von Ergebnissen abzurufen.

Paginatoren sind eine Funktion von AWS SDK für PHP, die als Abstraktion für diesen Prozess dienen, um Entwicklern die Verwendung von paginierten Seiten zu erleichtern. APIs Ein Paginator ist im Wesentlichen ein Iterator der Ergebnisse. Sie werden über die `getPaginator()` -Methode des Clients erstellt. Wenn Sie `getPaginator()` aufrufen, müssen Sie den Namen der Operation und die Argumente der Operation angeben (auf dieselbe Art und Weise wie bei der Ausführung einer Operation). Sie können mit `foreach` ein Paginator-Objekt durchlaufen, um einzelne `Aws\Result`-Objekte zu erhalten.

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

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

Paginator-Objekte

Das von der `getPaginator()`-Methode zurückgegebene Objekt ist eine Instance der `Aws\ResultPaginator`-Klasse. Diese Klasse implementiert die eigene `iterator`-Schnittstelle von PHP, weshalb sie mit `foreach` arbeitet. Sie kann auch mit Iterator-Funktionen verwendet werden, z. B. `iterator_to_array`, und kann problemlos mit [SPL-Iteratoren](#) wie dem `LimitIterator`-Objekt kombiniert werden.

Paginator-Objekte enthalten nur eine „Seite“ der Ergebnisse gleichzeitig und werden langsam ausgeführt. Das bedeutet, dass sie nur so viele Anfragen ausführen, wie erforderlich sind, um die aktuelle Ergebnisseite zu füllen. Beispielsweise gibt der Amazon S3 `ListObjects` S3-Vorgang nur

bis zu 1.000 Objekte gleichzeitig zurück. Wenn Ihr Bucket also ~10.000 Objekte enthält, müsste der Paginator insgesamt 10 Anfragen bearbeiten. Wenn Sie die Ergebnisse durchlaufen, wird die erste Anforderung ausgeführt, wenn Sie die Iteration starten, die zweite in der zweiten Iteration der Schleife usw.

Aufzählen von Daten aus Ergebnissen

Paginator-Objekte haben eine Methode namens `search()`, mit der Sie Iteratoren für Daten innerhalb einer Ergebnismenge erstellen können. Geben Sie beim Aufrufen einen [JMESPath Ausdruck](#) an `search()`, um anzugeben, welche Daten extrahiert werden sollen. Der Aufruf von `search()` gibt einen Iterator zurück, der die Ergebnisse des Ausdrucks auf jeder Ergebnisseite ausgibt. Dies wird langsam ausgewertet, weil Sie den zurückgegebenen Iterator durchlaufen.

Das folgende Beispiel entspricht dem vorherigen Codebeispiel, aber verwendet die `ResultPaginator::search()`-Methode, die kürzer ist.

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

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

JMESPath Ausdrücke ermöglichen es Ihnen, ziemlich komplexe Dinge zu tun. Wenn Sie z. B. alle Objektschlüssel und allgemeinen Präfixe ausgeben möchten (d. h. einen ls eines Buckets), können Sie wie folgt vorgehen.

```
// 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";
}
```

Asynchrone Paginierung

Sie können die Ergebnisse eines Paginators asynchron durchlaufen, indem Sie einen Callback für die `each()`-Methode eines `Aws\ResultPaginator` bereitstellen. Der Callback wird für jeden Wert aufgerufen, der vom Paginator geliefert wird.

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

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

Note

Mit der `each()`-Methode können Sie die Ergebnisse einer API-Operation paginieren, während gleichzeitig asynchron weitere Abfragen gesendet werden.

Ein Rückgabewert ungleich Null aus dem Callback wird durch das zugrundeliegende, auf einer Co-Routine basierende Promise erzielt. Das bedeutet, dass Sie Promises aus dem Rückruf, die aufgelöst werden müssen, zurückgeben können, bevor Sie die Iteration über die verbleibenden Positionen fortsetzen und im Wesentlichen in andere Promises zur Iteration übergehen. Der letzte Wert ungleich Null, der von dem Callback zurückgegeben wird, ist das Ergebnis, das das Promise aller nachgelagerten Promises erfüllt. Wenn der letzte Rückgabewert ist ein Promise ist, ist die Auflösung dieses Promise das Ergebnis, das die nachgelagerten Promises erfüllt oder ablehnt.

```
// 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();
```

Kellner in der AWS SDK für PHP Version 3

Waiter helfen dabei, mit eventuell konsistenten Systemen zu arbeiten, indem sie abstrahiert warten, bis eine Ressource in einen bestimmten Zustand gelangt, indem sie die Ressource abfragt. Eine Liste der von einem Client unterstützten Kellner finden Sie in der [API-Dokumentation](#) für eine einzelne Version eines Service-Clients. Um dorthin zu gelangen, gehen Sie zur Kundenseite in der API-Dokumentation und navigieren Sie zur spezifischen Versionsnummer (dargestellt durch ein Datum) und scrollen Sie nach unten zum Abschnitt „Kellner“. [Über diesen Link gelangen Sie zum Bereich Kellner von S3.](#)

Im folgenden Beispiel wird der Amazon S3 S3-Client verwendet, um einen Bucket zu erstellen. Dann wartet der Waiter, bis der Bucket existiert.

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

Wenn der Waiter den Bucket zu oft abfragen muss, wird eine `\RuntimeException` Ausnahme ausgelöst.

Konfiguration für Kellner

Waiter werden von einem assoziativen Array von Konfigurationsoptionen gesteuert. Alle von einem bestimmten Waiter verwendeten Optionen haben Standardwerte, aber sie können außer Kraft gesetzt werden, um verschiedene Wartestrategien zu unterstützen.

Sie können die Waiter-Konfigurationsoptionen ändern, indem Sie ein assoziatives Array von `@waiter` -Optionen an das `$args` -Argument der `waitUntil()`- und `getWaiter()`-Methoden eines Clients übergeben.

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

Verzögerung (int)

Anzahl der Sekunden, die zwischen den Sendeversuchen verzögert werden. Jeder Waiter hat einen Standardkonfigurationswert `delay`, aber Sie müssen diese Einstellung möglicherweise für bestimmte Anwendungsfälle ändern.

maxAttempts (int)

Maximale Anzahl von Sendeversuchen, die vor dem Fehlschlagen des Waiters ausgeführt werden sollen. Diese Option stellt sicher, dass Sie nicht unbegrenzt auf eine Ressource warten. Jeder Waiter hat einen Standardkonfigurationswert `maxAttempts`, aber Sie müssen diese Einstellung möglicherweise für bestimmte Anwendungsfälle ändern.

initDelay (int)

Zeit in Sekunden, die vor dem ersten Abrufversuch gewartet wird. Dies kann nützlich sein, wenn Sie auf eine Ressource warten, von der Sie wissen, dass sie eine Weile dauern wird, um in den gewünschten Zustand zu gelangen.

vor der Aktualisierung (abrufbar)

Eine abrufbare PHP-Funktion, die vor jedem Versuch aufgerufen wird. Das `abrufbar` wird mit dem Befehl `Aws\CommandInterface`, der gerade ausgeführt wird, und der Anzahl der bisher ausgeführten Versuche aufgerufen. Die Verwendung der aufrufbaren `before` könnte darin bestehen, Befehle zu modifizieren, bevor sie ausgeführt werden, oder Fortschrittsinformationen zur Verfügung zu stellen.

```
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
            );
        }
    ]
]);

```

Asynchron warten

Zusätzlich zum synchronen Warten können Sie einen Waiter auffordern, asynchron zu warten, während er andere Anfragen sendet oder auf mehrere Ressourcen gleichzeitig wartet.

Sie können auf ein WaiterPromise zugreifen, indem Sie einen Waiter von einem Client mithilfe der `getWaiter($name, array $args = [])`-Methode des Clients abrufen. Verwenden Sie die `promise()`-Methode eines Waiters, um den Waiter zu initiieren. Ein WaiterPromise wird mit der letzten `Aws\CommandInterface` erfüllt, die im Waiter ausgeführt wurde, und mit einer `RuntimeException` im Fehlerfall abgelehnt.

```

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();
```

Die Bereitstellung einer auf Promise basierenden Waiter-API ermöglicht einige leistungsstarke und relativ geringe Overhead-Anwendungsfälle. Zum Beispiel, was ist, wenn Sie auf mehrere Ressourcen warten und etwas mit dem ersten Waiter machen möchten, der erfolgreich gelöst wurde?

```
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 Ausdrücke in der AWS SDK für PHP Version 3

[JMESPath](#) ermöglicht es Ihnen, deklarativ anzugeben, wie Elemente aus einem JSON-Dokument extrahiert werden sollen. Das AWS SDK für PHP ist von [jmespath.php](#) abhängig, um einige der abstrakten Ebenen wie [Paginators in der AWS SDK für PHP Version 3 und Waiters in der Version 3 zu unterstützen, macht aber auch die AWS SDK für PHP Suche](#) nach und verfügbar. JMESPath `Aws\ResultInterface` `Aws\ResultPaginator`

Sie können JMESPath in Ihrem Browser damit herumspielen, indem Sie die [Online-Beispiele ausprobieren. JMESPath](#). In der [JMESPath Spezifikation](#) können Sie mehr über die Sprache, einschließlich der verfügbaren Ausdrücke und Funktionen, erfahren.

Die [AWS CLI](#) stützt JMESPath. Ausdrücke, die Sie für die CLI-Ausgabe schreiben, sind zu 100 Prozent mit Ausdrücken kompatibel, die für die Datei AWS SDK für PHP geschrieben wurden.

Extrahieren von Daten aus Ergebnissen

Die `Aws\ResultInterface` Schnittstelle verfügt über eine `search($expression)` Methode, mit der Daten aus einem Ergebnismodell auf der Grundlage eines JMESPath Ausdrucks extrahiert werden. Die Verwendung von JMESPath Ausdrücken zur Abfrage der Daten aus einem Ergebnisobjekt kann dazu beitragen, bedingten Standardcode zu entfernen und die Daten, die extrahiert werden, präziser auszudrücken.

Um zu demonstrieren, wie das funktioniert, beginnen wir unten mit der Standard-JSON-Ausgabe, die zwei Amazon Elastic Block Store (Amazon EBS) -Volumes beschreibt, die an separate EC2 Amazon-Instances angehängt sind.

```
$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"
    }
}
}

```

Zuerst können wir mit dem folgenden Befehl nur das erste Volume aus der Volumes-Liste angezeigt aufrufen.

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

Nun benutzen wir den wildcard-indexAusdruck[*], um über die gesamte Liste zu iterieren und auch drei Elemente zu extrahieren und umzubenennen: VolumeId wird in ID umbenannt,

AvailabilityZone wird in AZ umbenannt und Size bleibt Size. Wir können diese Elemente extrahieren und umbenennen, indem wir einen Ausdruck multi-hash verwenden, der hinter dem Ausdruck wildcard-index steht.

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

Dies gibt uns eine Reihe von PHP-Daten wie folgt:

```
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)
  }
}
```

In der multi-hash Notation können Sie auch verkettete Schlüssel wie `key1.key2[0].key3` verwenden, um tief geschachtelte Elemente innerhalb der Struktur zu filtern. Das Beispiel unten zeigt dies anhand des Schlüssels `Attachments[0].InstanceId` für den der einfache Alias `InstanceId` verwendet wird. (In den meisten Fällen ignorieren JMESPath Ausdrücke Leerzeichen.)

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

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

Der vorherige Ausdruck gibt folgende Daten aus:

```
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)
  }
}
```

Sie können auch mehrere Elemente mit dem `multi-list` Ausdruck `[key1, key2]` filtern: Dadurch werden alle gefilterten Attribute in einer einzelnen geordneten Liste pro Objekt formatiert, unabhängig vom Typ.

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

Das Ausführen der vorherigen Suche erzeugt die folgenden Daten:

```
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)
}
}

```

Verwenden Sie einen Ausdruck `filter`, um die Ergebnisse nach dem Wert eines bestimmten Feldes zu filtern. Die folgende Beispielabfrage gibt nur Volumes in der Availability Zone `us-west-2a` aus.

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

JMESPath unterstützt auch Funktionsausdrücke. Nehmen wir an, Sie möchten dieselbe Abfrage wie oben ausführen, aber stattdessen alle Volumes abrufen, in denen sich das Volume in einer AWS Region befindet, die mit „us-“ beginnt. Der folgende Ausdruck verwendet die Funktion `starts_with` und übergibt ein String-Literal von `us-`. Das Ergebnis dieser Funktion wird dann mit dem JSON-Literalwert von `true` verglichen, wobei nur die Ergebnisse des Filterprädikats weitergegeben werden, das `true` durch die Filterprojektion zurückgegeben hat.

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

Extrahieren von Daten aus Paginatoren

Wie Sie aus den [Paginatoren im Handbuch zu AWS SDK für PHP Version 3](#) wissen, werden `Aws\ResultPaginator` Objekte verwendet, um Ergebnisse aus einer API-Operation mit Seitenzugriff zu erhalten. AWS SDK für PHP Damit können Sie gefilterte Daten aus `Aws\ResultPaginator`

Objekten extrahieren und über sie iterieren. Dabei wird im Wesentlichen eine [Flatmap](#) über dem Iterator implementiert, bei der das Ergebnis eines Ausdrucks die Map-Funktion ist JMESPath .

Nehmen wir an, Sie möchten eine `iterator` erstellen, die nur Objekte aus einem Bucket liefert, die größer als 1 MB sind. Dies kann erreicht werden, indem zuerst ein `ListObjectsUmbruch` erzeugt wird und dann eine `search()` -Funktion auf den Umbruch angewendet wird, wodurch ein Flat-Mapped-Iterator über den paginierten Daten erzeugt wird.

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

// The result yielded as $data will be each individual match from
// Contents in which the Size attribute is > 1048576
foreach ($filtered as $data) {
    var_dump($data);
}
```

AWS-Services Von der AWS SDK für PHP Version 3 aus aufrufen

Die folgenden Abschnitte enthalten Beispiele, Tutorials, Aufgaben und Anleitungen, die Ihnen zeigen, wie Sie mit AWS Diensten arbeiten können. AWS SDK für PHP

Themen

- [Funktionen und Optionen der AWS SDK für PHP Version 3 nutzen](#)
- [Codebeispiele mit Anleitungen für die AWS SDK für PHP](#)

Funktionen und Optionen der AWS SDK für PHP Version 3 nutzen

Die AWS SDK für PHP Version 3 bietet Unterstützung für zusätzliche Funktionen und Optionen, mit denen Sie arbeiten können AWS-Service APIs. In den Abschnitten dieses Themas werden diese Optionen nach Diensten geordnet behandelt.

Themen

- [Verwenden des DynamoDB-Sitzungshandlers mit Version 3 AWS SDK für PHP](#)
- [Funktionen und Optionen von Amazon S3 in AWS SDK für PHP Version 3](#)

Verwenden des DynamoDB-Sitzungshandlers mit Version 3 AWS SDK für PHP

Der DynamoDB Session Handler ist ein benutzerdefinierter Session-Handler für PHP, der es Entwicklern ermöglicht, Amazon DynamoDB als Sitzungsspeicher zu verwenden. Die Verwendung von DynamoDB als Sitzungsspeicher behebt Probleme, die bei der Sitzungsverarbeitung in einer verteilten Webanwendung auftreten, indem Sitzungen aus dem lokalen Dateisystem an einen gemeinsam genutzten Speicherort verschoben werden. DynamoDB ist schnell, skalierbar, einfach einzurichten und wickelt die Replikation Ihrer Daten automatisch ab.

Der DynamoDB-Sitzungshandler verwendet die `session_set_save_handler()` Funktion, um DynamoDB-Operationen in die [systemeigenen Sitzungsfunktionen](#) von PHP einzubinden, um einen echten Ersatz zu ermöglichen. Dies beinhaltet die Unterstützung von Funktionen wie Sitzungssperren und Speicherbereinigung, die Teil des standardmäßigen Sitzungshandlers von PHP sind.

Weitere Informationen zum DynamoDB-Service finden Sie auf der [Amazon DynamoDB DynamoDB-Homepage](#).

Grundlegende Verwendung

Schritt 1: Registrieren Sie den Handler

Zuerst instanziiieren und registrieren Sie den Sitzungs-Handler.

```
use Aws\DynamoDb\SessionHandler;

$dynamoDb = new Aws\DynamoDb\DynamoDbClient([
    'region'=>'us-east-1' // Since version 3.277.10 of the SDK,
]);                      // the 'version' parameter defaults to 'latest'.

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

$sessionHandler->register();
```

Schritt 2. Erstellen Sie eine Tabelle zum Speichern Ihrer Sitzungen

Bevor Sie den Sitzungs-Handler tatsächlich verwenden können, müssen Sie eine Tabelle erstellen, in der die Sitzungen gespeichert werden. Sie können dies im Voraus tun, indem Sie die [AWS Konsole für Amazon DynamoDB](#) verwenden oder die AWS SDK für PHP

Beim Erstellen dieser Tabelle wählen Sie „id“ als Name des Primärschlüssels. Außerdem wird empfohlen, ein [Time to Live-Attribut](#) einzurichten, und zwar anhand des Attributs „expires“. So profitieren Sie von der Sitzungsspeicherbereinigung.

Schritt 3. Verwenden Sie PHP-Sitzungen wie gewohnt

Sobald der Sitzungshandler registriert ist und die Tabelle existiert, können Sie mit der superglobalen Variable `$_SESSION` in die Sitzung schreiben und aus ihr lesen, genau wie Sie es normalerweise mit dem Standard-Sitzungshandler von PHP tun. Der DynamoDB Session Handler kapselt und abstrahiert die Interaktionen mit DynamoDB und ermöglicht es Ihnen, einfach die nativen Sitzungsfunktionen und die Schnittstelle von PHP zu verwenden.

```
// 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();
```

Konfiguration

Sie können das Verhalten des Sitzungs-Handlers mithilfe der folgenden Optionen konfigurieren. Alle Optionen sind optional, aber Sie sollten sicher sein, dass Sie die Standardeinstellungen kennen.

table_name

Der Name der DynamoDB-Tabelle, in der die Sitzungen gespeichert werden sollen. Der Standardwert ist 'sessions'.

hash_key

Der Name des Hash-Schlüssels in der DynamoDB-Sitzungstabelle. Der Standardwert ist 'id'.

data_attribute

Der Name des Attributs in der DynamoDB-Sitzungstabelle, in der die Sitzungsdaten gespeichert sind. Der Standardwert ist 'data'.

data_attribute_type

Der Typ des Attributs in der DynamoDB-Sitzungstabelle, in der die Sitzungsdaten gespeichert sind. Dies ist standardmäßig 'string', kann aber optional als 'binary' festgelegt werden.

session_lifetime

Die Lebensdauer einer inaktiven Sitzung, bevor der Speicher bereinigt werden soll. Wenn sie nicht angegeben ist, wird `ini_get('session.gc_maxlifetime')` als Wert für die Lebensdauer verwendet.

session_lifetime_attribute

Der Name des Attributs in der DynamoDB-Sitzungstabelle, in dem die Ablaufzeit der Sitzung gespeichert ist. Der Standardwert ist 'expires'.

consistent_read

Ob der Sitzungs-Handler konsistente Lesevorgänge für die `GetItem`-Operation verwenden soll. Der Standardwert ist `true`.

locking

Ob Sitzungssperren verwendet werden sollen. Der Standardwert ist `false`.

batch_config

Konfiguration zum Stapellöschen bei der Speicherbereinigung. Diese Optionen werden direkt an [DynamoDB-Objekte `WriteRequestBatch`](#) übergeben. Lösen Sie die Speicherbereinigung manuell über `SessionHandler::garbageCollect()` aus.

max_lock_wait_time

Maximale Zeit (in Sekunden), wie lange der Sitzungs-Handler warten soll, bis ihm eine Sperre erteilt wird, bevor er aufgibt. Der Standardwert ist 10 und wird nur für die Sitzungssperre verwendet.

min_lock_retry_microtime

Minimale Zeit (in Mikrosekunden), wie lange der Sitzungs-Handler zwischen den Versuchen warten soll, eine Sperre zu erhalten. Der Standardwert ist 10000 und wird nur für die Sitzungssperre verwendet.

max_lock_retry_microtime

Maximale Zeit (in Mikrosekunden), wie lange der Sitzungs-Handler zwischen den Versuchen warten soll, eine Sperre zu erhalten. Der Standardwert ist 50000 und wird nur für die Sitzungssperre verwendet.

Zur Konfiguration des Session Handlers geben Sie bei der Instanziierung des Handlers die Konfigurationsoptionen an. Der folgende Code ist ein Beispiel mit allen Konfigurationsoptionen.

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

```

Preisgestaltung

Abgesehen von den Gebühren für Datenspeicherung und Datenübertragung werden die mit der Nutzung von DynamoDB verbundenen Kosten auf der Grundlage der bereitgestellten Durchsatzkapazität Ihrer Tabelle berechnet (siehe Preisdetails zu [Amazon DynamoDB](#)). Der Durchsatz wird in Einheiten der Schreib- und Lesekapazität gemessen. Auf der Amazon DynamoDB DynamoDB-Homepage heißt es:

Eine Lesekapazitätseinheit entspricht einem Strongly Consistent-Lesevorgang pro Sekunde oder zwei Eventually Consistent-Lesevorgängen pro Sekunde für Elemente mit einer Größe von bis zu 4 KB. Eine Schreibkapazitätseinheit entspricht einem Schreibvorgang pro Sekunde für Elemente von bis zu 1 KB.

Letztendlich korrelieren der Durchsatz und die Kosten für Ihre Sitzungstabelle mit dem erwarteten Datenverkehr und der Sitzungsgröße. In der folgenden Tabelle wird die Anzahl der Lese- und Schreiboperationen erläutert, die in Ihrer DynamoDB-Tabelle für jede der Sitzungsfunktionen ausgeführt werden.

Lesen über `session_start()`

- 1 Lesevorgang (nur 0,5, wenn `consistent_read` gleich `false` ist).
- (Bedingt) 1 Schreibvorgang zum Löschen der Sitzung, falls sie abgelaufen ist.

Lesen über `session_start()` (mit Sitzungssperre)

- Mindestens 1 Schreibvorgang.
- (Bedingt) Zusätzliche Schreibvorgänge für jeden Versuch, eine Sitzungssperre zu erhalten. Basierend auf der konfigurierte Wartezeit für die Sperre und Wiederholungsoptionen.
- (Bedingt) 1 Schreibvorgang zum Löschen der Sitzung, falls sie abgelaufen ist.

Schreiben über <code>session_write_close()</code>	• 1 Schreibvorgang.
Löschen über <code>session_destroy()</code>	• 1 Schreibvorgang.
Speicherbereinigung	• 0,5 Lesevorgänge pro 4 KB Daten in der Tabelle, um nach abgelaufenen Sitzungen zu suchen. • 1 Schreibvorgang pro abgelaufenem Element, um dieses zu löschen.

Sperren von Sitzungen

Der DynamoDB Session Handler unterstützt pessimistisches Sperren von Sitzungen, um das Verhalten des Standard-Session-Handlers von PHP nachzuahmen. Standardmäßig ist diese Funktion im DynamoDB-Sitzungshandler deaktiviert, da sie zu einem Leistungsengpass werden und die Kosten in die Höhe treiben kann, insbesondere wenn eine Anwendung über Ajax-Anfragen oder Iframes auf die Sitzung zugreift. Überlegen Sie sorgfältig, ob Ihre Anwendung eine Sitzungssperre benötigt, bevor Sie diese aktivieren.

Um die Sitzungssperre zu aktivieren, setzen Sie die `'locking'`-Option auf `true`, wenn Sie den `SessionHandler` instanziiieren.

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

Müllabfuhr

Richten Sie in Ihrer DynamoDB-Tabelle mit dem Attribut „expires“ ein TTL-Attribut ein. Dies bereinigt Ihre Sitzungen automatisch und erspart Ihnen, sie selbst entfernen zu müssen.

Alternativ unterstützt der DynamoDB-Sitzungshandler die Sitzungsbereinigung mithilfe einer Reihe von Scan Und-Operationen. `BatchWriteItem` Aufgrund der Art, wie die Scan-Operation funktioniert, und um alle abgelaufenen Sitzungen zu finden und zu löschen, kann der Speicherbereinigungsprozess eine Menge Durchsatz erfordern.

Aus diesem Grund unterstützen wir keine automatisierte Speicherbereinigung. Eine bessere Praxis ist es, die Speicherbereinigung so zu planen, dass sie außerhalb der Hauptverkehrszeiten

stattfindet, in der eine Erhöhung des verbrauchten Durchsatzes den Rest der Anwendung nicht stört. Beispielsweise könnte ein nächtlicher Cron-Job ein Skript zur Ausführung der Speicherbereinigung auslösen. Dieses Skript müsste etwa Folgendes erledigen.

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

$sessionHandler->garbageCollect();
```

Sie können auch die 'before'-Option mit 'batch_config' verwenden, um Verzögerungen für die BatchWriteItem-Operationen anzuwenden, die von der Speicherbereinigung ausgeführt werden. Dadurch verlängert sich die Zeit, die für die Garbage-Collection benötigt wird, aber es kann Ihnen helfen, die Anfragen des DynamoDB-Sitzungshandlers so zu verteilen, dass Sie während der Garbage-Collection in der Nähe oder innerhalb Ihrer bereitgestellten Durchsatzkapazität bleiben.

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

$sessionHandler->garbageCollect();
```

Best Practices

1. Erstellen Sie Ihre Sitzungstabelle in einer AWS Region, die Ihren Anwendungsservern geografisch am nächsten liegt oder sich in derselben Region wie diese befindet. Dadurch wird die niedrigste Latenz zwischen Ihrer Anwendung und der DynamoDB-Datenbank gewährleistet.
2. Wählen Sie die bereitgestellte Durchsatzkapazität für Ihre Sitzungstabelle sehr sorgfältig aus. Berücksichtigen Sie den erwarteten Datenverkehr zu Ihrer Anwendung und die erwartete Größe

- Ihrer Sitzungen. Verwenden Sie alternativ den Read/Write Kapazitätsmodus „On Demand“ für Ihre Tabelle.
- Überwachen Sie Ihren verbrauchten Durchsatz über die AWS Management Console oder mit Amazon CloudWatch und passen Sie Ihre Durchsatzeinstellungen nach Bedarf an die Anforderungen Ihrer Anwendung an.
 - Halten Sie die Größe Ihrer Sitzungen klein (im Idealfall weniger als 1 KB). Kleine Sitzungen sind leistungsfähiger und erfordern weniger Durchsatzkapazität.
 - Verwenden Sie keine Sitzungssperren, es sei denn, Ihre Anwendung benötigt sie.
 - Anstatt die eingebauten Auslöser für die Speicherbereinigung von Sitzungen in PHP zu verwenden, planen Sie Ihre Speicherbereinigung über einen Cron-Job oder einen anderen Scheduling-Mechanismus so ein, dass sie außerhalb der Stoßzeiten stattfindet. Nutzen Sie die 'batch_config'-Option.

Erforderliche IAM-Berechtigungen

Um DynamoDB verwenden zu können SessionHandler, müssen Ihre [konfigurierten Anmeldeinformationen](#) berechtigt sein, die DynamoDB-Tabelle zu verwenden, die [Sie in einem vorherigen Schritt erstellt](#) haben. Die folgende IAM-Richtlinie enthält die Mindestberechtigungen, die Sie benötigen. Um diese Richtlinie zu verwenden, ersetzen Sie den Ressourcenwert durch den Amazon-Ressourcennamen (ARN) der Tabelle, die Sie zuvor erstellt haben. Weitere Informationen zum Erstellen und Anhängen von IAM-Richtlinien finden Sie unter [Verwaltung von IAM-Richtlinien](#) im IAM-Benutzerhandbuch.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SessionHandler",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:Scan",
        "dynamodb:BatchWriteItem"
      ],
      "Effect": "Allow",

```

```
"Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-  
name"  
  }  
]  
}
```

Funktionen und Optionen von Amazon S3 in AWS SDK für PHP Version 3

In diesem Thema werden zusätzliche Funktionen und Optionen beschrieben, die AWS SDK für PHP Version 3 für die Verwendung mit Amazon S3 bietet.

Themen

- [Amazon S3 S3-Client mit mehreren Regionen in AWS SDK für PHP Version 3](#)
- [Amazon S3 S3-Stream-Wrapper in der AWS SDK für PHP Version 3](#)
- [Dateien und Verzeichnisse übertragen](#)
- [Clientseitige Amazon S3 S3-Verschlüsselung in Version 3 AWS SDK für PHP](#)
- [Schutz der Datenintegrität mit Prüfsummen](#)

Amazon S3 S3-Client mit mehreren Regionen in AWS SDK für PHP Version 3

Die AWS SDK für PHP Version 3 bietet einen generischen Client für mehrere Regionen, der mit jedem Dienst verwendet werden kann. Auf diese Weise können Benutzer angeben, AWS in welche Region ein Befehl gesendet werden soll, indem sie einen `@region` Eingabeparameter für einen beliebigen Befehl angeben. Darüber hinaus bietet das SDK einen multiregionalen Client für Amazon S3, der intelligent auf bestimmte Amazon S3 S3-Fehler reagiert und Befehle entsprechend umleitet. Dadurch können Benutzer den gleichen Client verwenden, um mit mehreren Regionen zu kommunizieren. Dies ist eine besonders nützliche Funktion für Benutzer des [Amazon S3 Stream Wrappers mit AWS SDK für PHP Version 3](#), deren Buckets sich in mehreren Regionen befinden.

Grundlegende Verwendung

Das grundlegende Nutzungsmuster eines Amazon S3 S3-Clients ist dasselbe, unabhängig davon, ob ein Standard-S3-Client oder sein regionsübergreifendes Gegenstück verwendet wird. Der einzige Nutzungsunterschied auf Befehlsebene besteht darin, dass eine AWS Region mithilfe des `@region` Eingabeparameters angegeben werden kann.

```
// 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

Wenn Sie den Amazon S3 S3-Client für mehrere Regionen verwenden, werden Sie nicht auf permanente Weiterleitungsausnahmen stoßen. Ein standardmäßiger Amazon S3 S3-Client löst eine Instanz von `\Aws\S3\Exception\PermanentRedirectException`, wenn ein Befehl in die falsche Region gesendet wird. Ein Multi-Region-Client leitet den Befehl stattdessen an die richtige Region zurück.

Cache für die Bucket-Region

Amazon S3 S3-Clients mit mehreren Regionen verwalten einen internen Cache der AWS Regionen, in denen sich die jeweiligen Buckets befinden. Standardmäßig hat jeder Client seinen eigenen In-Memory-Cache. Wenn Sie einen Cache zwischen Clients oder Prozessen freigeben möchten, stellen Sie Ihrem Multi-Region-Client eine Instanz von `\Aws\CacheInterface` als Option `bucket_region_cache` zur Verfügung.

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

Amazon S3 S3-Stream-Wrapper in der AWS SDK für PHP Version 3

Der Amazon S3-Stream-Wrapper ermöglicht Ihnen das Speichern und Abrufen von Daten aus Amazon S3 mithilfe integrierter PHP-Funktionen wie `file_get_contents`, `fopen`, `copy`, `rename`, `unlink`, `mkdir`, `undrmdir`.

Sie müssen den Amazon S3 S3-Stream-Wrapper registrieren, um ihn verwenden zu können.

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

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

Auf diese Weise können Sie mithilfe des `s3://` Protokolls auf Buckets und Objekte zugreifen, die in Amazon S3 gespeichert sind. Der Amazon S3 S3-Stream-Wrapper akzeptiert Zeichenketten, die einen Bucket-Namen gefolgt von einem Schrägstrich und einem optionalen Objektschlüssel oder Präfix enthalten: `s3://<bucket>[/<key-or-prefix>]`

Note

Der Stream-Wrapper ist auf die Arbeit mit Objekten und Buckets ausgelegt, für die Sie mindestens Leseberechtigung besitzen. Das bedeutet, Ihr Benutzer benötigt die Berechtigung zum Ausführen von `ListBucket` für alle Buckets und `GetObject` für alle Objekte, mit denen die Benutzer interagieren muss. Für Anwendungsfälle, in denen Sie nicht über diese Berechtigungsstufe verfügen, empfehlen wir, dass Sie Amazon S3 S3-Client-Operationen direkt verwenden.

Daten herunterladen

Sie können den Inhalt eines Objekts mit `file_get_contents` abrufen. Seien Sie jedoch vorsichtig mit dieser Funktion; sie lädt den gesamten Inhalt des Objekts in den Speicher.

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

Verwenden Sie diese `fopen()` Option, wenn Sie mit größeren Dateien arbeiten oder Daten von Amazon S3 streamen müssen.

```
// 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

Datei-Schreibfehler werden nur dann zurückgegeben, wenn ein Aufruf von `fflush` durchgeführt wurde. Diese Fehler werden nicht zurückgegeben, wenn `fclose` ohne Leeren aufgerufen wurde. Der Rückgabewert für `fclose` ist `true`, wenn es den Stream schließt, unabhängig von Fehlern in der Antwort auf seinen internen `fflush`. Diese Fehler werden aufgrund der Implementierung durch PHP auch beim Aufruf von `file_put_contents` nicht zurückgegeben.

Öffnen Sie durchsuchbare Streams

Im „r“-Modus geöffnete Streams erlauben nur das Lesen von Daten aus dem Stream und sind standardmäßig nicht durchsuchbar. Auf diese Weise können Daten auf echte Streaming-Weise von Amazon S3 heruntergeladen werden, wobei zuvor gelesene Bytes nicht im Speicher zwischengespeichert werden müssen. Wenn Sie einen Stream brauchen, der durchsuchbar ist, können Sie `seekable` in den [Stream-Kontext-Optionen](#) einer Funktion übergeben.

```
$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);  
}
```

Das Öffnen von durchsuchbaren Streams ermöglicht es Ihnen, Bytes zu suchen, die zuvor gelesen wurden. Sie können nicht zu Bytes springen, die noch nicht vom Remote-Server gelesen wurden. Damit zuvor gelesene Daten abgerufen werden können, werden die Daten in einem temporären PHP-Stream mit Hilfe eines Stream-Decorators gepuffert. Wenn die Menge der zwischengespeicherten Daten 2 MB überschreitet, werden die Daten im temporären Datenstrom vom Speicher auf die Festplatte übertragen. Denken Sie daran, wenn Sie große Dateien von Amazon S3 mithilfe der seekable Stream-Kontext-Einstellung herunterladen.

Laden Sie Daten hoch

Sie können Daten mit auf Amazon S3 hochladen `file_put_contents()`.

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

Größere Dateien können Sie durch Streaming von Daten mit `fopen()` und dem Stream-Zugriffsmodus „w“, „x“ oder „a“ streamen. Der Amazon S3 S3-Stream-Wrapper unterstützt keine gleichzeitigen Lese- und Schreibstreams (z. B. „r+“, „w+“ usw.). Der Grund hierfür ist, dass das HTTP-Protokoll gleichzeitige Lese- und Schreibvorgänge nicht unterstützt.

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

Note

Amazon S3 erfordert die Angabe eines Content-Length-Headers, bevor die Nutzdaten einer Anfrage gesendet werden. Daher werden die Daten, die in einer PutObject-Operation hochgeladen werden sollen, intern mit einem temporären PHP-Stream gepuffert, bis der Stream geleert oder geschlossen wird.

 Note

Datei-Schreibfehler werden nur dann zurückgegeben, wenn ein Aufruf von `fflush` durchgeführt wurde. Diese Fehler werden nicht zurückgegeben, wenn `fclose` ohne Leeren aufgerufen wurde. Der Rückgabewert für `fclose` ist `true`, wenn es den Stream schließt, unabhängig von Fehlern in der Antwort auf seinen internen `fflush`. Diese Fehler werden aufgrund der Implementierung durch PHP auch beim Aufruf von `file_put_contents` nicht zurückgegeben.

Fopen-Modi

Die [fopen\(\)](#)-Funktion von PHP erfordert, dass Sie eine `$mode`-Option angeben. Die `mode`-Option legt fest, ob Daten in einen Stream geschrieben oder daraus gelesen werden können und ob die Datei beim Öffnen eines Streams vorhanden sein muss.

Der Amazon S3 S3-Stream-Wrapper unterstützt die folgenden Modi für Streams, die auf Amazon S3 S3-Objekte abzielen.

r	Ein schreibgeschützter Stream, bei dem das Objekt bereits existieren muss.
w	Ein Stream, in den nur geschrieben wird. Wenn das Objekt bereits existiert, wird es überschrieben.
a	Ein Stream, in den nur geschrieben wird. Wenn das Objekt bereits existiert, wird es in einen temporären Stream heruntergeladen, und alle Schreibvorgänge in den Stream werden an alle zuvor hochgeladenen Daten angehängt.
x	Ein Stream, in den nur geschrieben wird. Ein Fehler wird ausgelöst, wenn das Objekt bereits existiert.

Andere Objektfunktionen

Stream-Wrapper ermöglichen es vielen verschiedenen integrierten PHP-Funktionen, mit einem benutzerdefinierten System wie Amazon S3 zu arbeiten. Hier sind einige der Funktionen, die Sie mit dem Amazon S3-Stream-Wrapper mit in Amazon S3 gespeicherten Objekten ausführen können.

unlink()

Ein Objekt aus einem Bucket löschen.

```
// Delete an object from a bucket
unlink('s3://bucket/key');
```

Sie können alle verfügbaren Optionen an die Operation `DeleteObject` übergeben, um zu ändern, wie das Objekt gelöscht wird (z. B. Angabe einer bestimmten Objektversion).

```
// Delete a specific version of an
object from a bucket
unlink('s3://bucket/key', stream_co
ntext_create([
    's3' => ['VersionId' => '123']
]));
```

filesize()

Die Größe eines Objekts ermitteln.

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

is_file()

Prüft, ob eine URL eine Datei ist.

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

file_exists()

Prüft, ob ein Objekt vorhanden ist.

```
if (file_exists('s3://bucket/key'))
{
```


	<pre>echo 'It exists!'; }</pre>
<code>filetype()</code>	Prüft, ob eine URL einer Datei oder einen Bucket (dir) ist.
<code>file()</code>	Lädt den Inhalt eines Objekts in die Zeilen eines Arrays. Sie können alle verfügbaren Optionen an die Operation <code>GetObject</code> übergeben, um zu ändern, wie das Objekt heruntergeladen wird.
<code>filemtime()</code>	Ermittelt das letzte Änderungsdatum eines Objekts.
<code>rename()</code>	Benennt ein Objekt um, indem das Objekt kopiert und dann das Original gelöscht wird. Sie können die verfügbaren Optionen für die Operationen <code>CopyObject</code> und <code>DeleteObject</code> an die Stream-Kontextparameter übergeben, um zu ändern, wie das Objekt kopiert und gelöscht wird.

Note

Funktioniert zwar `copy` im Allgemeinen mit dem Amazon S3 S3-Stream-Wrapper, einige Fehler werden jedoch aufgrund der internen `copy` Funktionsweise der Funktion in PHP möglicherweise nicht richtig gemeldet. Wir empfehlen, stattdessen eine Instanz von [ObjectCopierAWSS3](#) zu verwenden.

Arbeiten Sie mit Buckets und Ordnern

Wird verwendet `mkdir()`, um mit Buckets zu arbeiten

Sie können Amazon S3 S3-Buckets auf ähnliche Weise erstellen und durchsuchen, wie PHP es Ihnen ermöglicht, Verzeichnisse in Ihrem Dateisystem zu erstellen und zu durchsuchen.

Hier ist ein Beispiel, das einen Bucket erstellt.

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

Note

Im April 2023 aktivierte Amazon S3 Block Public Access automatisch und deaktivierte Zugriffskontrolllisten für alle neu erstellten Buckets. Diese Änderung wirkt sich auch darauf aus, wie die StreamWrapper `mkdir` Funktion mit Berechtigungen und ACLs funktioniert. Weitere Informationen finden Sie in diesem [AWS Artikel Was gibt's Neues mit](#).

Sie können Stream-Kontext-Optionen an die `mkdir()` Methode übergeben, um mithilfe der für den [CreateBucket](#) Vorgang verfügbaren Parameter zu ändern, wie der Bucket erstellt wird.

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

Mit der `rmdir()`-Funktion können Sie Buckets löschen.

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

Note

Ein Bucket kann nur gelöscht werden, wenn er leer ist.

Wird verwendet **`mkdir()`**, um mit Ordnern zu arbeiten

Nachdem Sie einen Bucket erstellt haben, können Sie `mkdir()` damit Objekte erstellen, die wie in einem Dateisystem als Ordner fungieren.

Der folgende Codeausschnitt fügt dem vorhandenen Bucket mit dem Namen „amzn-s3-demo-bucket“ ein Ordnerobjekt mit dem Namen „my-folder“ hinzu. Verwenden Sie den Schrägstrich (/), um den Namen eines Ordnerobjekts vom Bucket-Namen und jedem weiteren Ordnernamen zu trennen.

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

Der [vorherige Hinweis](#) zu Berechtigungsänderungen nach April 2023 gilt auch für das Erstellen von Ordnerobjekten. [Dieser Blogbeitrag](#) enthält Informationen darüber, wie Sie Berechtigungen bei Bedarf anpassen können.

Verwenden Sie die `rmdir()` Funktion, um ein leeres Ordnerobjekt zu löschen, wie im folgenden Codeausschnitt gezeigt.

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

Listet den Inhalt eines Buckets auf

Sie können die PHP-Funktionen [opendir\(\)](#), [readdir\(\)](#), [rewinddir\(\)](#) und [closedir\(\)](#) mit dem Amazon S3 S3-Stream-Wrapper verwenden, um den Inhalt eines Buckets zu durchsuchen. Sie können Parameter, die für den [ListObjects](#) Vorgang verfügbar sind, als benutzerdefinierte Stream-Kontextoptionen an die Funktion übergeben, um die Art und Weise zu ändern, wie Objekte aufgelistet werden. `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);
}
```

Mithilfe von PHP können Sie jedes Objekt und jedes Präfix in einem Bucket rekursiv auflisten.

[RecursiveDirectoryIterator](#)

```
$dir = 's3://bucket';
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir));

foreach ($iterator as $file) {
    echo $file->getType() . ': ' . $file . "\n";
}
```

Eine andere Möglichkeit, den Inhalt eines Buckets rekursiv aufzulisten, der weniger HTTP-Anfragen erzeugt, ist die Funktion `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";
}
```

Optionen für den Stream-Kontext

Sie können den vom Stream-Wrapper verwendeten Client oder den Cache, in dem zuvor geladene Informationen über Buckets und Schlüssel zwischengespeichert werden, anpassen, indem Sie benutzerdefinierte Stream-Kontext-Optionen übergeben.

Der Stream-Wrapper unterstützt die folgenden Stream-Kontext-Optionen für jede Operation.

client

Das `Aws\AwsClientInterface`-Objekt zum Ausführen von Befehlen.

cache

Eine Instance von `Aws\CacheInterface` für das Caching zuvor ermittelter Dateistatistiken. Standardmäßig verwendet der Stream-Wrapper einen In-Memory-LRU-Cache.

Dateien und Verzeichnisse übertragen

Die AWS SDK für PHP Version 3 bietet zwei Ansätze für die Übertragung von Dateien und Verzeichnissen zu und von Amazon S3. Beide Lösungen bewältigen die Komplexität von mehrteiligen Uploads und Downloads großer Dateien, unterscheiden sich jedoch in ihrer Designphilosophie, ihrem Funktionsumfang und ihren Nutzungsmustern.

Übersicht über die Übertragungsoptionen

Wählen Sie den Übertragungsansatz, der den Anforderungen Ihrer Anwendung am besten entspricht:

S3 Transfer Manager (empfohlen)

Eine moderne Bibliothek auf hohem Niveau, die eine umfassende Lösung für Dateiübertragungen bietet. Sie bietet umfangreiche Konfigurationsoptionen, eine integrierte Fortschrittsverfolgung, benutzerdefinierte Download-Handler und eine robuste Fehlerbehandlung. Der S3 Transfer Manager verwendet eine auf Zusagen basierende API und unterstützt sowohl einzelne Dateioperationen als auch Verzeichnisübertragungen mit erweiterten Filterfunktionen.

Übertragung

Eine Implementierung der Verzeichnisübertragung, die sich speziell auf Massenverzeichnisoperationen konzentriert. Sie bietet eine einfachere API zum Hoch- und Herunterladen ganzer Verzeichnisse mit grundlegenden Konfigurationsoptionen. Dieser Ansatz bietet im Vergleich zum S3 Transfer Manager weniger Funktionen.

Die wichtigsten Unterschiede:

In der folgenden Tabelle werden die Hauptunterschiede zwischen den beiden Übertragungsansätzen hervorgehoben:

Feature	S3-Übertragungsmanager	Übertragung
Einzelne Dateioperationen	Ja (einzelne Dateien hochladen/herunterladen)	Nein (nur Verzeichnisoperationen)
Verzeichnisoperationen	Ja (mit erweiterter Filterung)	Ja (grundlegende Verzeichnisübertragungen)
Verfolgung des Fortschritts	Eingebaut mit benutzerdefinierten Listenern	Eingeschränkt (nur Debug-Ausgabe)
Benutzerdefinierte Download-Handler	Ja	Nein
Überprüfung der Prüfsumme	Automatisch mit Konfiguration	Manuell (Option <code>add_content_md5</code>)
Fehlerbehandlung	Umfassend mit Fehlerrichtlinien	Grundlegendes, auf Versprechen basierendes Handling

Feature	S3-Übertragungsmanager	Übertragung
Konfigurationsoptionen	Umfangreich (mehr als 8 Optionen)	Grundlegend (6 Optionen)
API-Design	Anforderungs-/Antwortobjekte	Einfache Konstruktorparameter

Den richtigen Ansatz wählen

Verwenden Sie den S3 Transfer Manager, wenn Sie:

- Upload- oder Download-Vorgänge für einzelne Dateien
- Fortschrittliche Verfolgung und Überwachung des Fortschritts
- Benutzerdefinierte Download-Handler für spezielle Verarbeitungen
- Umfassende Richtlinien für die Fehlerbehandlung und Wiederholungsversuche
- Präzise Kontrolle über mehrteilige Operationen
- Verzeichnisoperationen mit komplexer Filterlogik

Verwenden Sie Transfer, wenn Sie Folgendes benötigen:

- Einfache Übertragungen von Verzeichnis zu und von S3
- Minimale Konfiguration und Einrichtung
- Kompatibilität mit vorhandenem Code, der Transfer verwendet
- Grundlegende mehrteilige Upload-Funktionalität

Note

Für neue Anwendungen empfehlen wir die Verwendung des S3 Transfer Managers, da er eine umfassendere und flexiblere Lösung für Dateiübertragungen bietet.

S3-Übertragungsmanager

Der S3 Transfer Manager bietet eine Schnittstelle zum Hoch- und Herunterladen von Dateien zu und von Amazon S3. Sie können ihn für einzelne Dateioperationen oder Verzeichnisoperationen verwenden.

Der Transfer Manager verarbeitet mehrteilige Uploads und Downloads automatisch. Er verwaltet große Dateien und verfolgt den Fortschritt. Die Bibliothek implementiert bewährte Methoden für S3-Datenübertragungen. Sie können sie verwenden, um Dateiübertragungsfunktionen in Ihren PHP-Anwendungen zu erstellen.

Schlüssel-Features

Der S3 Transfer Manager bietet die folgenden Hauptfunktionen:

- Einfache API: Dateien und Verzeichnisse hoch- und herunterladen
- Benutzerdefinierte Download-Handler: Implementieren Sie Ihre eigene Download-Logik
- Automatische mehrteilige Uploads und Downloads: Behandeln Sie große Dateien automatisch
- Gleichzeitige Verarbeitung: Maximieren Sie den Durchsatz
- Fortschrittsverfolgung: Überwachen Sie den Übertragungsstatus
- Anpassbares Verhalten: Konfigurieren Sie umfangreiche Optionen
- Fehlerbehandlung: Konfigurieren Sie Richtlinien für Wiederholungen und Fehler
- Verzeichnisoperationen: Übertragen Sie mehrere Dateien stapelweise
- Prüfsummenvalidierung: Stellen Sie die Datenintegrität sicher

Installation

Der S3 Transfer Manager ist in der AWS SDK für PHP Version 3 enthalten. Sie müssen ihn nicht separat installieren.

Führen Sie zur Installation mit Composer den folgenden Befehl aus:

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

Dieser Befehl installiert die komplette AWS SDK für PHP Version 3, einschließlich des S3 Transfer Managers.

Grundlegende Verwendung

Das folgende Beispiel zeigt, wie der S3 Transfer Manager verwendet wird:

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

// Create a transfer manager with default configuration.
$transferManager = new S3TransferManager($s3Client);

// Alternative: Create transfer manager with null client. S3 Transfer Manager uses a
// default S3 client.
$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Example: Upload a file.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/local/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'path/to/s3/file.txt',
        ]
    )
);

// Wait for the upload to complete.
$result = $uploadPromise->wait();

echo "Upload complete!\n";
```


⚠ Important

Wenn Sie S3 Transfer Manager verwenden, um einen standardmäßigen Amazon S3 S3-Client zu erstellen, können Kunden mithilfe des `default_region` Parameters in den Transfer config Manager-Optionen eine Standardregion für den Client übergeben. Andernfalls versucht der Amazon S3 S3-Client, eine Region mithilfe des Standardverhaltens für die Auflösung von Konfigurationen aufzulösen, und wenn die Region nicht aufgelöst wird, wird eine Ausnahme ausgelöst.

Einen Transfermanager erstellen

Sie können einen Transfermanager auf zwei Arten erstellen:

Mit einem vorhandenen S3-Client

Übergeben Sie eine bestehende [S3Client](#)-Instanz an den `S3TransferManager` *<add link>* Konstruktor.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\S3TransferManager;

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

// Create an S3 client.
$s3Client = new S3Client([
    'version' => 'latest',
    'region'  => 'us-west-2',
]);

$transferManager = new S3TransferManager($s3Client);
```

Mit standardmäßiger S3-Client-Erstellung

Übergeben Sie `null` es als Client und geben Sie die Konfigurationsoptionen an.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;
```

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);
```

Konfiguration

S3 Transfer Manager akzeptiert Konfigurationsoptionen, um sein Verhalten anzupassen. Geben Sie diese Optionen an, wenn Sie eine Instanz des Transfer Managers erstellen. Der Konfigurationsparameter kann entweder ein Array oder eine Instanz von sein `S3TransferManagerConfig`[<add link>](#).

Im folgenden Beispiel wird eine S3 Transfer Manager-Instanz konfiguriert:

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null,
    [
        // 10MB parts for multipart operations.
        'target_part_size_bytes' => 10 * 1024 * 1024,

        // Use multipart upload for files larger than 20MB.
        'multipart_upload_threshold_bytes' => 20 * 1024 * 1024,

        // Enable checksum calculation for data integrity.
        'request_checksum_calculation' => 'when_supported',

        // Enable checksum validation when getting objects.
        'response_checksum_validation' => 'when_supported',

        // Use part-based multipart downloads.
        'multipart_download_type' => 'part',

        // Allow up to 10 concurrent operations.
        'concurrency' => 10,
```

```
// Enable progress tracking.
'track_progress' => true,

// Set default region for default S3 client construction.
'default_region' => 'us-west-2',
]
);
```

Note

Wenn Sie die Konfiguration als Array angeben `S3TransferManager`, ruft das SDK intern auf, `S3TransferManagerConfig::fromArray` um sie in den richtigen Typ zu konvertieren.

Konfigurationsoptionen

Alle Konfigurationsoptionen sind optional und verwenden Standardwerte, wenn sie nicht angegeben sind.

Option	Typ	Standard	Description
<code>target_part_size_bytes</code>	int	8 MB	Die Mindestteilgröße für mehrteilige Uploads/Downloads.
<code>multipart_upload_threshold_bytes</code>	int	16 MB	Schwellenwert für die Dateigröße für den mehrteiligen Upload.
<code>request_checksum_calculation</code>	Zeichenfolge	'wenn_unterstützt'	Aktiviert die Prüfsummenberechnung. Gültige Werte sind 'when_supported', 'when_required'.
<code>response_checksum_validation</code>	Zeichenfolge	'wenn_unterstützt'	Aktiviert die Prüfsummenvalidierung beim Abrufen von Objekten. Gültige Werte sind 'when_supported', 'when_required'.

Option	Typ	Standard	Description
<code>multipart_download_type</code>	Zeichenfolge	„Teil“	Download-Strategie für große Dateien. Gültige Werte sind „Teil“ (mehrteiliger Download), „Bereich“ (Bereichsanforderungen).
<code>concurrency</code>	int	5	Maximale Anzahl gleichzeitiger Operationen.
<code>track_progress</code>	bool	FALSE	Ob der Übertragungsfortschritt verfolgt werden soll.
<code>default_region</code>	Zeichenfolge	'US-Ost-1'	AWS-Region zu verwenden, wenn kein S3-Client bereitgestellt wird.

Dateioperationen

S3 Transfer Manager bietet Methoden zum Hoch- und Herunterladen einzelner Dateien.

Laden Sie lokale Dateien hoch

Verwenden Sie die `upload` [<add link>](#) Methode, um eine Datei auf Amazon S3 hochzuladen.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Source can be from a local path to a file.
$source = '/path/to/local/file.txt';
// Or the source can be an instance of StreamInterface.
$source = GuzzleHttp\Psr7\Utils::streamFor('Hello World!');

$uploadPromise = $transferManager->upload(
```

```
new UploadRequest(
    $source,
    [
        'Bucket' => 'amzn-s3-demo-bucket',
        'Key'     => 'path/to/s3/file.txt',
        // Additional `putObject` parameters as needed.
    ],
    [
        // Optional configuration overrides.
        'multipart_upload_threshold_bytes' => 100 * 1024 * 1024, // 100MB
        'target_part_size_bytes'          => 10 * 1024 * 1024, // 10MB
        'track_progress'                  => true,
        'request_checksum_calculation'    => 'when_required',
    ]
)
);

// The upload is asynchronous, you can wait for it to complete.
$result = $uploadPromise->wait();

// Or you can use the promise for more complex workflows.
$result = $uploadPromise->then(
    function ($result) {
        echo "Upload succeeded!";
    },
    function ($error) {
        echo "Upload failed: " . $error->getMessage();
    }
)->wait();
```

Parameter für die **upload**-Methode

Die upload Methode akzeptiert eine Instanz von UploadRequest [<add link>](#) als Argument.

UploadRequest-Parameter

Parameter	Typ	Erforderlich	Description
\$source	Zeichenkette StreamInterface	Ja	Pfad des lokalen Dateipfads oder Streams, der die hochzuladenden Daten enthält.

Parameter	Typ	Erforderlich	Description
<code>\$uploadRequestArgs</code>	Array	Ja	Argumente für die Upload-Anforderung (müssen Bucket und Key enthalten).
<code>\$config</code>	Array	Nein	Spezifische Konfigurationsüberschreibungen für diesen Upload. Weitere Informationen zu den Konfigurationsoptionen finden Sie im folgenden Abschnitt .
<code>\$listeners</code>	Array	Nein	Array von <code>TransferListener</code> Objekten zur Überwachung dieses Uploads.
<code>\$progressTracker</code>	<code>TransferListener</code>	Nein	Ein Fortschritts-Tracker für diesen Upload.

`$config` Optionen für `UploadRequest`

Das SDK löst den `$config` Standardwert aus der [Konfiguration](#) des S3 Transfer Managers auf.

Option	Typ	Erforderlich	Description
<code>multipart_upload_threshold_bytes</code>	int	Nein	Setzt den Standardschwellenwert für den Fall außer Kraft, dass ein mehrteiliger Upload ausgelöst wird.
<code>target_part_size_bytes</code>	int	Nein	Setzt die standardmäßige Zielteileröße in Byte außer Kraft.
<code>track_progress</code>	bool	Nein	Überschreibt die Standardoption, um die Fortschrittsverfolgung zu aktivieren. Wenn diese Option aktiviert ist <code>true</code> und Sie keinen <code>progressT</code>

Option	Typ	Erforderlich	Description
			racke Parameter angeben, verwendet das SDK eine Standardimplementierung.
concurrency	int	Nein	Setzt den Standardwert für Parallelität außer Kraft.
request_checksum_calculation	Zeichenfolge	Nein	Setzt den Standardwert für die Angabe außer Kraft, ob eine Prüfsummenberechnung für eine Anfrage durchgeführt werden muss.

Wenn die upload Methode erfolgreich ausgeführt wird, gibt sie eine zurück. UploadResult [<add link>](#)

Laden Sie S3-Objekte herunter

Verwenden Sie die download [<add link>](#) Methode, um ein Objekt von Amazon S3 herunterzuladen.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

// Download using an S3 URI.
$downloadPromise = $transferManager->download(
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
        [
            // Additional `getObject` parameters as needed.
        ],
        [
```

```
// Optional configuration overrides.
'response_checksum_validation' => 'when_required',
'track_progress'               => true,
'target_part_size_bytes'       => 10 * 1024 * 1024, // 10MB
    ]
)
);

// Wait for the download to complete.
$result = $downloadPromise->wait();

// The SDK uses a stream-based download handler by default.
$stream = $result->getDownloadDataResult();

// You can either get the content.
$content = $stream->getContents();

// Or write the stream to a file.
file_put_contents('/path/to/local/file.txt', $stream);

// Don't forget to close the stream.
$stream->close();
```

Parameter für die **download**-Methode

Die `download` Methode akzeptiert eine Instanz von `DownloadRequest` [<add link>](#) als Argument.

DownloadRequest-Parameter

Parameter	Typ	Erforderlich	Description
<code>\$source</code>	Zeichenkette Matrix <code>null</code>	Nein	Das Objekt, das von S3 heruntergeladen werden soll. Sie können diesen Parameter als S3-URI-Zeichenfolge (" s3://amzn-s3-demo-bucket/key „), als Array mit Bucket- und Key-Schlüsseln oder angeben <code>null</code> . Wenn dieser Wert lautet <code>null</code> , übergeben Sie die Bucket- und Key-Werte im

Parameter	Typ	Erforderlich	Description
			<code>\$downloadRequestArgs</code> Parameter.
<code>\$downloadRequestArgs</code>	Array	Nein	Zusätzliche Argumente für die Anforderung von Download-Objekten. Falls ja <code>\$sourcenu11</code> , geben Sie hier die Bucket- und Key-Werte an.
<code>\$config</code>	Array	Nein	Spezifische Konfigurationsüberschreibungen für diesen Download. Weitere Informationen zu den Konfigurationsoptionen finden Sie im folgenden Abschnitt .

Parameter	Typ	Erforderlich	Description
<code>\$downloadHandler</code>	<code>AbstractDownloadHandler</code> <i><add link></i> <code> null</code>	Nein	Der Handler, der jeden Objektblock empfängt und dessen Download verwaltet. Der Standard-Handler für die <code>download</code> Methode verwendet einen Stream-basierten Handler. Dieser Handler schiebt jeden Objekt-Chunk in einen Stream. Der Standard-Handler für die <code>downloadFile</code> Methode (siehe the section called “Laden Sie ein S3-Objekt in eine lokale Datei herunter”) verwendet einen dateibasierten Handler. Dieser Handler schreibt jeden Chunk in eine Datei. Sie können Ihre eigenen implementieren <code>DownloadHandler</code>, um anzupassen, wo und wie das SDK heruntergeladene Daten speichert. Sie können beispielsweise Daten in Datenbanken, Cloud-Speichern oder benutzerdefinierten Verarbeitungspipelines statt in Dateien oder Streams speichern.
<code>\$progressTracker</code>	<code>TransferListener</code>	Nein	Ein Fortschritts-Tracker für diesen Download.
<code>\$listeners</code>	<code>array</code>	Nein	Reihe von <code>AbstractTransferListener</code> Objekten zur Überwachung dieses Downloads.

\$config Optionen für **DownloadRequest**

Das SDK löst den \$config Standardwert aus der [Konfiguration](#) des S3 Transfer Managers auf.

Option	Typ	Erforderlich	Beschreibung
<code>multipart_download_type</code>	Zeichenfolge	Nein	Setzt den standardmäßigen mehrteiligen Downloadtyp außer Kraft. Gültige Werte sind „Teil“, „Bereich“
<code>response_checksum_validation</code>	Zeichenfolge	Nein	Überschreibt den in der Transfer-Manager-Konfiguration aufgelösten Wert, der angibt, ob die Prüfsummenvalidierung durchgeführt werden soll. Das SDK berücksichtigt diese Option nur, wenn sie nicht in der <code>\$getObjectRequestArgs</code> von vorhanden <code>ChecksumMode</code> ist. <code>DownloadRequest</code>
<code>track_progress</code>	bool	Nein	Setzt die Standardoption für die Aktivierung der Fortschrittsverfolgung außer Kraft. Wenn diese Option aktiviert ist <code>true</code> und Sie keinen <code>progressTracker</code> Parameter angeben, verwendet <code>DownloadRequest</code>, das SDK eine Standardimplementierung. Verwenden Sie diese Option, um einen Standard-Fortschritts-Tracker zu aktivieren, wenn der <code>\$progressTracker</code> Parameter lautet <code>null</code>.
<code>target_part_size_bytes</code>	int	Nein	Die Größe des Teils in Byte, das bei einem mehrteiligen Range-Download verwendet werden soll. Wenn Sie diesen Parameter nicht angeben,

Option	Typ	Erforderlich	Beschreibung
			verwendet der Download den im Transfer Manager <code>target_part_size_bytes</code> konfigurierten Parameter.

Wenn die `download` Methode erfolgreich ausgeführt wird, gibt sie eine zurück `DownloadResult` [<add link>](#).

Laden Sie ein S3-Objekt in eine lokale Datei herunter

Verwenden Sie die folgende `downloadFile` Methode, um ein S3-Objekt direkt in eine lokale Datei herunterzuladen:

```
<?php

use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$downloadFilePromise = $transferManager->downloadFile(
    new DownloadFileRequest(
        '/path/to/local/file.txt', // Destination file path.
        false, // Fail when destination exists.
        new DownloadRequest(
            's3://amzn-s3-demo-bucket/path/to/s3/file.txt',
            [], // `getObject` request args
            [
                'track_progress' => true,
                'target_part_size_bytes' => 10 * 1024 * 1024, // 10MB parts
            ]
        )
    )
);
```

```
// Wait for download to complete.
$result = $downloadFilePromise->wait();

// `getDownloadDataResult()` returns the string for the file path of the downloaded
// content because
// the default `DownloadHandler` used by `downloadFile()` is a file-based handler.
echo "File downloaded successfully at {$result->getDownloadDataResult()}!\n";
```

Parameter für die **downloadFile**-Methode

Die `downloadFile` Methode akzeptiert eine Instanz von `DownloadFileRequest` als Argument.

DownloadFileRequest-Parameter

Option	Typ	Erforderlich	Beschreibung
<code>\$destination</code>	Zeichenfolge	Ja	Lokaler Pfad, in dem die Datei gespeichert ist.
<code>\$failWhenDestinationExists</code>	bool	Ja	Ob ein Fehler auftreten soll, wenn die Zielfeile bereits existiert. Wenn diese Option aktiviert ist <code>false</code> und die Zielfeile existiert, löscht oder überschreibt das SDK die vorhandene Datei.
<code>\$downloadRequest</code>	<code>DownloadRequest</code>	Ja	Das konfigurierte Objekt <code>DownloadRequest</code> . Weitere Informationen finden Sie unter DownloadRequest Objekt .

Wenn die `downloadFile` Methode erfolgreich ausgeführt wird, gibt sie ein zurück `DownloadResult` [<add link>](#).

Verzeichnisoperationen

S3 Transfer Manager kann ganze Verzeichnisübertragungen abwickeln.

Verzeichnis hochladen

S3 Transfer Manager kann ein ganzes Verzeichnis in einen S3-Bucket hochladen. *<Are the key values the filenames when uploaded?>*

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/local/directory',
        'amzn-s3-demo-bucket',
        [
            // Additional `putObject` parameters that apply to all files.
            'ACL' => 'public-read',
            'CacheControl' => 'max-age=3600',
        ],
        [
            // Configuration options.
            'recursive' => true,
            'follow_symbolic_links' => false,
            's3_prefix' => 'uploads/2023/',
            's3_delimiter' => '/',
            'track_progress' => true,
            'filter' => function ($file) {
                // Upload only .jpg files.
                return pathinfo($file, PATHINFO_EXTENSION) === 'jpg';
            },
            'upload_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ContentType'] = 'image/jpeg';
            },
        ]
    )
);
```

```
// Wait for the upload process to complete.
$result = $uploadDirPromise->wait();

$uploaded = $result->getObjectsUploaded();
$failed = $result->getObjectsFailed();
echo "Uploaded {$uploaded} files. Failed: {$failed}\n";
```

Parameter für die **uploadDirectory**-Methode

Die `uploadDirectory` Methode akzeptiert eine Instanz von `UploadDirectoryRequest` *<add link>* als Argument.

UploadDirectoryRequest-Parameter

Parameter	Typ	Erforderlich	Beschreibung
<code>\$sourceDirectory</code>	Zeichenfolge	Ja	Pfad des lokalen Verzeichnisses, das hochgeladen werden soll.
<code>\$targetBucket</code>	Zeichenfolge	Ja	Name des Ziel-S3-Buckets.
<code>\$uploadRequestArgs</code>	Array	Nein	Zusätzliche Upload-Objektanforderungsparameter für alle Dateien.
<code>\$config</code>	Array	Nein	Konfigurationsoptionen für den Verzeichnis-Upload. Weitere Informationen zu den Konfigurationsoptionen finden Sie im folgenden Abschnitt .
<code>\$listeners</code>	Array	Nein	Array von <code>TransferListener</code> - Objekten. Das SDK kloniert jedes Exemplar <code>TransferListener</code> pro Datei.
<code>\$max_depth</code>	bool	-1 „Standard laufzeit“	Um die maximale Tiefe der rekursiven Datei anzugeben, laufen drei Schritte.

Parameter	Typ	Erforderlich	Beschreibung
<code>\$progressTracker</code>	<code>TransferListener</code>	Nein	Ein Fortschritts-Tracker für alle Uploads.

\$configOptionen für **UploadDirectoryRequest**

Option	Typ	Standard	Description
<code>recursive</code>	<code>bool</code>	<code>false</code>	Ob Unterverzeichnisse rekursiv hochgeladen werden sollen.
<code>follow_symbolic_links</code>	<code>bool</code>	<code>false</code>	Ob symbolischen Links gefolgt werden soll.
<code>s3_prefix</code>	Zeichenfolge	<code>"</code>	Präfix, das allen Objektschlüsseln hinzugefügt werden soll.
<code>s3_delimiter</code>	Zeichenfolge	<code>'</code>	Das in S3-Objektschlüsseln zu verwendende Trennzeichen.
<code>track_progress</code>	<code>bool</code>	<code>false</code>	Ob der Fortschritt verfolgt werden soll.
<code>max_concurrency</code>	<code>int</code>	<code>100</code>	Die maximale Anzahl gleichzeitiger Uploads.
<code>filter</code>	aufrufbar	<code>Null</code>	Funktion zum Filtern der hochzuladenden Dateien.
<code>upload_object_request_modifier</code>	aufrufbar	<code>Null</code>	Funktion zum Anpassen jeder Upload-Anfrage.
<code>failure_policy</code>	aufrufbar	<code>Null</code>	Funktion zur Behandlung von Upload-Fehlern.

Aufrufbare Arten von Optionen **\$config**

Optionsname	Name des Parameters	Parametertyp	Parameter-Informationen
<code>filter</code>	<code>\$file</code>	<code>SplFileInfo</code> Zeichenfolge	Wenn es in eine Zeichenfolge umgewandelt wird, ist dies der Dateipfad. Andernfalls ist es eine Instanz von <code>SplFileInfo</code> .
<code>upload_object_request_modifier</code>	<code>\$requestArgs</code>	Array	Die Anforderungsargumente für jede einzelne Upload-Anfrage. Weitere Informationen zu Array-Optionen finden Sie im Abschnitt zur Parametersyntax der PutObject-Methode .
<code>failure_policy</code>	<code>\$uploadRequestArgs</code>	Array	Die Anforderungsargumente für jede einzelne Upload-Anfrage. Weitere Informationen zu Array-Optionen finden Sie im Abschnitt zur Parametersyntax der PutObject-Methode für Array-Optionen.
	<code>\$uploadDirectoryArgs</code>	Array	Ein Array mit dem Quellverzeichnis und dem Ziel-Bucket für den Vorgang

Optionsname	Name des Parameters	Parametertyp	Parameter-Informationen
			zum Hochladen des Verzeichnisses.
	\$reason	Throwable Zeichenfolge	Der Ausnahmehalter, der Informationen darüber enthält, was den Fehler verursacht hat.
	\$uploadDirectoryResult	UploadDirectoryResult	Das Objekt, das die Anzahl der erfolgreich hochgeladenen Objekte und die Anzahl der fehlgeschlagenen Objekte enthält.

Wenn die `uploadDirectory` Methode erfolgreich ausgeführt wird, gibt sie eine zurück `UploadDirectoryResult`[<add link>](#).

Verzeichnis herunterladen

Sie können ein Verzeichnis aus einem S3-Bucket herunterladen. *<Are the filenames the key values when downloaded? If there is no extension in the key, is there also no extension on the file?>*

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, [
    'default_region' => 'us-west-2'
]);
```

```

$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/local/directory',
        [
            // Additional `getObject` parameters that apply to all files.
        ],
        [
            // Configuration options.
            's3_prefix' => 'uploads/2023/',
            's3_delimiter' => '/',
            'track_progress' => true,
            'filter' => function ($key) {
                // Download only .jpg files.
                return pathinfo($key, PATHINFO_EXTENSION) === 'jpg';
            },
            'download_object_request_modifier' => function ($args) {
                // Customize request arguments for each file.
                $args['ResponseContentType'] = 'image/jpeg';
            },
            'list_objects_v2_args' => [
                'MaxKeys' => 1000,
                'Prefix' => 'uploads/2023/',
            ],
            'fails_when_destination_exists' => true,
        ]
    )
);

// Wait for the download process to complete.
$result = $downloadDirPromise->wait();

$downloaded = $result->getObjectsDownloaded();
$failed = $result->getObjectsFailed();
echo "Downloaded {$downloaded} files. Failed: {$failed}\n";

```

Parameter für die **downloadDirectory**-Methode

Die `downloadDirectory` Methode akzeptiert eine Instanz von `DownloadDirectoryRequest` [<add Link>](#) als Argument.

DownloadDirectoryRequest-Parameter

Parameter	Typ	Erforderlich	Beschreibung
<code>\$sourceBucket</code>	Zeichenfolge	Ja	Name des Quell-S3-Buckets, aus dem Objekte heruntergeladen werden.
<code>\$destinationDirectory</code>	Zeichenfolge	Ja	Lokales Verzeichnis, in das Dateien heruntergeladen werden.
<code>\$downloadRequestArgs</code>	Array	Nein	Zusätzliche Parameter für die Anforderung von Download-Objekten für alle Dateien.
<code>\$config</code>	Array	Nein	Konfigurationsoptionen für den Verzeichnis-Download. Weitere Informationen zu den Konfigurationsoptionen finden Sie im folgenden Abschnitt .
<code>\$listeners</code>	Array	Nein	Array von <code>TransferListener</code> - Objekten. Das SDK kloniert jedes Exemplar <code>TransferListener</code> pro Datei.
<code>\$progressTracker</code>	<code>TransferListener</code>	Nein	Ein Fortschritts-Tracker für alle Downloads.

\$configOptionen für **DownloadDirectoryRequest**

Option	Typ	Standard	Description
<code>s3_prefix</code>	Zeichenfolge	"	Präfix für die Filterung (falls nicht <code>list_objects_v2_args</code> vorhanden).

Option	Typ	Standard	Description
<code>track_progress</code>	<code>bool</code>	<code>false</code>	Ob der Fortschritt verfolgt werden soll.
<code>filter</code>	aufrufbar	<code>null</code>	Funktion zum Filtern, welche S3-Objekte heruntergeladen werden sollen.
<code>download_object_request_modifier</code>	aufrufbar	<code>null</code>	Funktion zum Anpassen jeder Download-Anfrage.
<code>list_objects_v2_args</code>	Array	<code>[]</code>	Argumente für die ListObjectsV2 Operation. Bei diesem Vorgang werden die Metadaten für die herunterzuladenden Objekte abgerufen.
<code>fails_when_destination_exists</code>	<code>bool</code>	<code>false</code>	Gibt an, ob ein Fehler auftreten soll, wenn ein Zielpfad für das Objekt bereits vorhanden ist.
<code>failure_policy</code>	aufrufbar	<code>Null</code>	Funktion zur Behandlung von Download-Fehlern.
<code>max_concurrency</code>	<code>int</code>	<code>100</code>	Die maximale Anzahl gleichzeitiger Downloads.
<code>target_part_size_bytes</code>	<code>int</code>	variiert	Teilgröße für mehrteilige Downloads, wenn der Typ des mehrteiligen Downloads in einem Bereich liegt.

Aufrufbare Typen von Optionen **\$config**

Optionsname	Name des Parameters	Parametertyp	Parameter-Informationen
<code>filter</code>	<code>\$objectKey</code>	Zeichenfolge	Der Name des Objektschlüssels aus dem Bucket.
<code>download_object_request_modifier</code>	<code>\$requestArgs</code>	Array	Die Anforderungsargumente für jede einzelne Download-Anfrage. Weitere Informationen zu Array-Optionen finden Sie im Abschnitt zur Parametersyntax der GetObject-Methode .
<code>failure_policy</code>	<code>\$downloadObjectRequestArgs</code>	Array	Die Anforderungsargumente für jede einzelne Download-Anfrage. Weitere Informationen zu Array-Optionen finden Sie im Abschnitt zur Parametersyntax der GetObject-Methode .
	<code>\$downloadDirectoryRequest</code>	Array	Ein Array mit dem Quell-Bucket und dem Zielverzeichnis für den Download-Verzeichnisvorgang.

Optionsname	Name des Parameters	Parametertyp	Parameter-Informationen
	<code>\$reason</code>	Throwable	Der Ausnahmehalter, der Informationen darüber enthält, was den Fehler verursacht hat.
	<code>\$downloadDirectoryResult</code>	DownloadDirectoryResult <add Link>	Das Objekt, das die Anzahl der erfolgreich heruntergeladenen Objekte und die Anzahl der fehlgeschlagenen Objekte enthält.

Wie funktionieren Verzeichnisoperationen

In diesem Abschnitt wird erklärt, wie der S3 Transfer Manager Dateipfade und Objektschlüssel bei Verzeichnisoperationen verarbeitet.

Behandlung von Upload-Pfaden

Wenn Sie ein Verzeichnis hochladen, erstellt das SDK S3-Objektschlüssel aus Dateipfaden:

1. Berechnet den Pfad jeder Datei relativ zum Quellverzeichnis
2. Stellt den `s3_prefix` Wert voran, falls angegeben (fügt automatisch einen abschließenden Schrägstrich hinzu)
3. Ersetzt das Betriebssystem-Verzeichnistrennzeichen durch das (Standard) `s3_delimiter /`

Die `recursive` Option steuert, ob Unterverzeichnisse eingeschlossen werden. Wenn `false` (Standard), werden nur Dateien der obersten Ebene hochgeladen. Wenn `true`, werden alle Dateien in Unterverzeichnissen hochgeladen, wobei die Verzeichnisstruktur in Objektschlüsseln beibehalten wird.

 Note

Der Schlüssel in S3 für jede hochgeladene Datei ist das angegebene s3-Präfix, standardmäßig ist es Null, plus der relative Pfad der Datei, beginnend mit dem angegebenen Quellverzeichnis. Außerdem ersetzen wir das Verzeichnistrennzeichen im aufgelösten Schlüsselnamen durch das bereitgestellte s3-Trennzeichen, das standardmäßig `/` ist.

Beispiel 1

```
$sourceDirectory = "/opt/my-upload-directory";  
$s3Prefix = "important/";  
$s3Delimiter = "/"; // Default value
```

Verzeichnisstruktur

```
/opt/my-upload-directory/  
-- my-file1.txt  
-- my-file-2.txt  
-- sub-dir/  
---- my-another-file1.txt  
---- my-another-file2.txt
```

Die Schlüssel für jede Datei lauten:

```
$s3Prefix + $fileRelativePath;
```

```
- important/my-file1.txt  
- important/my-file-2.txt  
- important/sub-dir/my-another-file1.txt  
- important/sub-dir/my-another-file2.txt
```

Beispiel 2

```
$sourceDirectory = "/opt/my-docs";  
$s3Prefix = ''; // No provided and it will defaulted to empty string  
$s3Delimiter = "/"; // Default value
```

Verzeichnisstruktur


```
/opt/my-docs/  
-- my-file1.txt  
-- my-file-2.txt  
-- sub-dir/  
---- my-another-file1.txt  
---- my-another-file2.txt
```

Die Schlüssel für jede Datei werden sein:

```
$s3Prefix + $fileRelativePath;
```

```
- my-file1.txt  
- my-file-2.txt  
- sub-dir/my-another-file1.txt  
- sub-dir/my-another-file2.txt
```

Example Verwaltung von Upload-Verzeichnispfaden

```
<?php  
// Source directory: /home/user/photos/  
//   fla/beach.jpg  
//   fla/sunset.jpg  
//   fla/pool/party.jpg  
// With s3_prefix: '2023' and recursive: true  
// Results:  
//   2023/fla/beach.jpg  
//   2023/fla/sunset.jpg  
//   2023/fla/pool/party.jpg
```

Verwaltung von Download-Pfaden

Wenn Sie ein Verzeichnis herunterladen, erstellt das SDK lokale Dateipfade aus S3-Objektschlüsseln:

1. Entfernt den `s3_prefix` aus dem Objektschlüssel, falls angegeben
2. Ersetzt das S3-Trennzeichen (`/`) durch das Verzeichnistrennzeichen des Betriebssystems

3. Fügt das Ergebnis an das Zielverzeichnis an

Downloads sind immer rekursiv. Das SDK ruft alle Objekte unter dem angegebenen Präfix ab und erstellt nach Bedarf Unterverzeichnisse. Aus Sicherheitsgründen überprüft es, dass Pfade nicht außerhalb des Zielverzeichnisses aufgelöst werden.

Example Verwaltung von Verzeichnispfaden herunterladen

```
<?php
// S3 objects:
// 2023/fla/beach.jpg
// 2023/fla/sunset.jpg
// 2023/fla/pool/party.jpg
// With s3_prefix: '2023' and destination: /home/user/downloads
// Results:
// /home/user/downloads/fla/beach.jpg
// /home/user/downloads/fla/sunset.jpg
// /home/user/downloads/fla/pool/party.jpg
```

Note

Der Dateipfad für die heruntergeladenen Objekte wird wie folgt aufgelöst:

- Das s3-Präfix, falls angegeben, wird aus dem Objektschlüssel entfernt.
- Wenn sich das s3Delimiter vom Betriebssystem-Verzeichnistrennzeichen unterscheidet, wird das s3-Trennzeichen durch das Betriebssystem-Verzeichnistrennzeichen ersetzt.

Beispiel 1

```
$s3Prefix = "my-prefix";
$s3Objects = [
    "my-prefix/file-1.txt",
    "my-prefix/file-2.txt",
    "my-prefix/subdir/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Sobald diese Objekte heruntergeladen wurden, werden sie wie folgt platziert:

```
/opt/bucket/downloads/
```

```
-- file-1.txt
-- file-2.txt
-- subdir/
---- file-1.txt
```

Wie wir sehen können, wurde das Präfix aus dem endgültigen Dateipfad für die Objekte entfernt.

Beispiel 2 - ohne s3-Präfix

```
$s3Prefix = ""; // No provided
$s3Objects = [
    "README.md",
    "my-docs/file-1.txt",
    "my-docs/file-2.txt",
    "my-docs/statements/file-1.txt"
];
$targetDirectory = "/opt/bucket/downloads/";
```

Sobald diese Objekte heruntergeladen wurden, werden sie wie folgt platziert:

```
/opt/bucket/downloads/
-- README.md
-- my-docs/
---- file-1.txt
---- file-2.txt
---- statements/
----- file-1.txt
```

Zuhörer übertragen

Sie können Ihren eigenen Listener implementieren, der auf verschiedene Übertragungsereignisse reagiert. Die Ereignisse sind:

- **transferInitiated**: Wenn eine Übertragung eingeleitet wird
- **bytesTransferred**: Wenn bei einer Übertragung Bytes von der Quelle zum Ziel übertragen werden
- **transferComplete**: Wenn eine Übertragung ohne Fehler abgeschlossen wird
- **transferFail**: Wenn eine Übertragung vor Abschluss fehlschlug

Jede `S3TransferManager` Operation (zum Beispiel `uploadFile` und `downloadDirectory`) hat einen `listeners` Parameter. Dieser Parameter ist ein Array, das erwartet, dass jedes Element eine Instanz von `AbstractTransferListener` ist.

`AbstractTransferListener` ist eine abstrakte Basisklasse, die Sie erweitern können, um Übertragungsvorgänge zu überwachen. Sie stellt leere Methodenimplementierungen für alle Übertragungsereignisse bereit. Sie müssen nur die Methoden für die Ereignisse überschreiben, die Sie behandeln möchten.

Jede `AbstractTransferListener` Methode erhält ein `$context` Argument. Dieses Argument enthält Schlüsselwertinformationen über den Vorgang. Der Kontext beinhaltet:

- `request_args`(Array) — Parameter anfordern
- `progress_snapshot`(Instanz von `TransferProgressSnapshot` [<add link>](#)) — Aktueller Anforderungsstatus
- `reason`(Instanz von `Throwable`) — wird nur für die `transferFail` Methode bereitgestellt

Die `progress_snapshot` Eigenschaft enthält auch die `reason` Eigenschaft für Fehlerereignisse. Sie können über `$context['reason']` oder über auf die Ausnahme zugreifen `$context['progress_snapshot']#getReason()`.

Der folgende Code zeigt die `AbstractTransferListener` Klassenstruktur:

```
<?php

namespace Aws\S3\S3Transfer\Progress;

abstract class AbstractTransferListener
{
    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the request initialization.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferInitiated(array $context): void {}

    /**
```

```

    * @param array $context
    * - request_args: (array) The request arguments that will be provided
    *   as part of the operation that originated the bytes transferred event.
    * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
    *
    * @return void
    */
    public function bytesTransferred(array $context): bool {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     *
     * @return void
     */
    public function transferComplete(array $context): void {}

    /**
     * @param array $context
     * - request_args: (array) The request arguments that will be provided
     *   as part of the operation that originated the bytes transferred event.
     * - progress_snapshot: (TransferProgressSnapshot) The transfer snapshot holder.
     * - reason: (Throwable) The exception originated by the transfer failure.
     *
     * @return void
     */
    public function transferFail(array $context): void {}
}

```

Das folgende Beispiel zeigt eine `AbstractTransferListener` Implementierung, die nur die `transferFail` Methoden `transferInitiated` und überschreibt:

```

<?php

namespace MyApp\Listeners;

use Aws\S3\S3Transfer\Progress\AbstractTransferListener;

class MyCustomListener extends AbstractTransferListener
{
    public function transferInitiated(array $context): void

```

```

{
    $progressSnapshot = $context['progress_snapshot'];

    echo "Transfer initiated for object with identifier: "
        . $progressSnapshot->getIdentifier();
}

public function transferFail(array $context): void
{
    $progressSnapshot = $context['progress_snapshot'];
    $reason = $context['reason'];

    echo "Transfer for object with identifier: "
        . $progressSnapshot->getIdentifier() . " has failed.";

    echo "Completion Status: " . $progressSnapshot->getTransferredBytes()
        . "/"
        . $progressSnapshot->getTotalBytes() . " B";

    echo "Reason of failure is: " . $reason;
}
}

```

```

<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use MyApp\Listeners\MyCustomListener;

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

// The following example uses `MyCustomListener` in an `upload` operation.
$transferManager = new S3TransferManager(
    null,
    ['default_region' => 'us-east-2']
);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        source: "/tmp/myfile.txt",
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'MyKey'

```

```
    ],
    listeners: [
        new MyCustomListener()
    ]
)
);

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

Nachverfolgung des Fortschritts

S3 Transfer Manager bietet integrierte Funktionen zur Fortschrittsverfolgung.

Verfolgung aller Operationen

Sie können die Nachverfolgung für alle Operationen (Datei und Verzeichnis) aktivieren, indem Sie `track_progress` `true` auf der `S3TransferManager` Instanz auf einstellen.

```
<?php

use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null,
    [
        'track_progress' => true,
    ]
);
```

Mit dieser Einstellung verwendet das SDK die integrierten Funktionen `SingleProgressTracker` [<add link>](#) für Dateioperationen und `MultiProgressTracker` [<add link>](#) für Verzeichnisoperationen.

Verfolgung eines einzelnen Dateivorgangs

Wenn Sie die Fortschrittsverfolgung im Transfer Manager nicht aktiviert haben, können Sie die Verfolgung für eine einzelne Methode auf zwei Arten aktivieren.

Wird `true` im `$config` Array für einen bestimmten Dateivorgang auf eingestellt `track_progress`.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [ // $config array
            'track_progress' => true,
        ],
        []
    )
);
```

Erstellen Sie eine neue `SingleProgressTracker` Instanz oder stellen Sie eine benutzerdefinierte Implementierung von `AbstractTransferListener`. Übergeben Sie Ihren Tracker als `$progressTracker` Parameter an die Dateioperationen.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/Myfile.txt', // Or an instance of StreamInterface.
    )
);
```



```

        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key' => 'file.txt',
        ],
        [],
        [],
        new SingleProgressTracker() // Or custom implementation of the
        TransferLister interface.
    )
);

```

Das folgende Beispiel zeigt eine Beispielausgabe der integrierten Konsole SingleProgressTracker für den Upload-Fortschritt einer Datei mit dem Namen MyFile.txt:

```

MyFile.txt:
[#####] 50% 5242880/10485760 B

```

Verfolgung eines einzelnen Verzeichnisvorgangs

Wenn Sie die Fortschrittsverfolgung im Transfer Manager nicht aktiviert haben, können Sie sie auf zwei Arten für Verzeichnisoperationen aktivieren.

Setzen Sie 'track_progress' true im \$config Array-Argument der Methode auf.

```

<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(
    null, []
);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [ // $config array

```

```
        'track_progress' => true,  
    ],  
    []  
)  
);
```

Erstellen Sie eine neue `MultiProgressTracker` Instanz und übergeben Sie sie als Parameter an die Verzeichnisoperationen. `$progressTracker`

```
<?php  
  
use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;  
use Aws\S3\S3Transfer\Progress\MultiProgressTracker;  
use Aws\S3\S3Transfer\S3TransferManager;  
  
require __DIR__ . '/../vendor/autoload.php';  
  
$transferManager = new S3TransferManager(  
    null, []  
);  
  
$uploadDirPromise = $transferManager->uploadDirectory(  
    new UploadDirectoryRequest(  
        '/path/to/directory',  
        'amzn-s3-demo-bucket',  
        [],  
        [],  
        [],  
        new MultiProgressTracker()  
    )  
);
```

Das folgende Beispiel zeigt eine Beispielausgabe der integrierten Konsole `MultiProgressTracker` für den Upload-Fortschritt eines Verzeichnisses:

```

MyFile.4.txt:
[#####] 100% 4194304/4194304 B
MyFile.5.txt:
[#####] 100% 24117248/24117248 B
MyFile.2.txt:
[#####] 0% 0/10485760 B
MyFile.3.txt:
[#####] 100% 18874368/18874368 B
MyFile.1.txt:
[#####] 100% 1048576/1048576 B
-----
[#####] 80% Completed: 3/5, Failed: 0/5

```

Der Fortschritt jeder Datei wird zusammen mit der Zusammenfassung des Fortschritts in der letzten Zeile aufgeführt.

Benutzerdefinierte Fortschrittsverfolgung

Sie können Ihre eigene Fortschrittsverfolgung für Dateioperationen implementieren. Erstellen Sie eine Klasse, die die `AbstractTransferListener` Schnittstelle erweitert, und übergeben Sie sie an die Methode in der Anfrage.

Die folgende Implementierung zeigt Fortschrittsinformationen im Textformat für einen einzelnen Dateivorgang an:

```

<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\AbstractTransferListener;
use Aws\S3\S3Transfer\S3TransferManager;

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

class MyProgressTracker extends AbstractTransferListener
{
    public function transferInitiated(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Starting transfer of {$key}...\n";
    }

    public function bytesTransferred(array $context): bool
    {
        $snapshot = $context['progress_snapshot'];
    }
}

```

```

        $percent = round($snapshot->ratioTransferred() * 100, 2);
        $transferred = round($snapshot->getTransferredBytes() / 1024 / 1024, 2);
        $total = round($snapshot->getTotalBytes() / 1024 / 1024, 2);

        echo "Progress: {$percent}% ({$transferred}MB of {$total}MB)\n";
    }

    public function transferComplete(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        echo "Transfer complete for {$key}!\n";
    }

    public function transferFail(array $context): void
    {
        $key = $context['request_args']['Key'] ?? 'unknown';
        $reason = $context['reason']->getMessage();
        echo "Transfer failed for {$key}: {$reason}\n";
    }
}

$transferManager = new S3TransferManager(
    null, []
);

// Use your custom tracker.
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt', // Or an instance of StreamInterface.
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'file.txt',
        ],
        [],
        [],
        new MyProgressTracker()
    )
);

```

Das folgende Beispiel zeigt eine Beispielausgabe an die Konsole für einen laufenden Datei-Upload:

```

Starting transfer of file.txt...
Progress: 42% (16.8MB of 40MB)

```

Passen Sie einen integrierten Fortschritts-Tracker an

Sie können die Fortschrittsverfolgung auf verschiedene Arten anpassen:

- Erstellen Sie eine benutzerdefinierte Implementierung, indem Sie sie erweitern `AbstractTransferListener` und implementieren `ProgressTrackerInterface` [<add link>](#).
- Passen Sie die vorhandenen Implementierungen an, indem Sie `ConsoleProgressBar` [<add link>](#) das konfigurieren, was an die integrierten Tracker übergeben wird.
- Implementieren Sie einen benutzerdefinierten `ProgressBar` [<add link>](#) Wert und übergeben Sie ihn an die integrierten Tracker.

Die folgenden Beispiele zeigen, wie Sie die Fortschrittsverfolgung anpassen können. Das [Klassendiagramm zeigt](#) die Beziehung zwischen den Komponenten zur Fortschrittsverfolgung im SDK.

Passen Sie eine an **ConsoleProgressBar**

Passen Sie das Standardbalkenformat an

Im folgenden Beispiel wird das Standardformat für Fortschrittsbalken angepasst, das ist `ColoredTransferProgressBarFormat` [<add link>](#). Dieses Format zeigt den Fortschritt je nach Übertragungsstatus in unterschiedlichen Farben an.

Das Beispiel ändert sich:

- das Zeichen von der Standardeinstellung „#“ zu „=“
- Balkenbreite von der Standardeinstellung 50 bis 100

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$progressBar = new ConsoleProgressBar(
```

```

        progressBarChar: '=',
        progressBarWidth: 100,
// Default `ColoredTransferProgressBarFormat` is used.
    );
    $progressTracker = new SingleProgressTracker(
        progressBar: $progressBar,
    );
    $s3Client = new S3Client([
        'region' => 'us-east-2',
    ]);
    $s3TransferManager = new S3TransferManager($s3Client);
    $source = "/path/file.txt";

    $result = $s3TransferManager->upload(
        new UploadRequest(
            source: $source,
            uploadRequestArgs: [
                'Bucket' => 'amzn-s3-demo-bucket',
                'Key'     => 'key',
            ],
            progressTracker: $progressTracker,
        )
    )->wait();
    print_r($result);

```

Die folgende Abbildung zeigt die Konsolenausgabe für ein benutzerdefiniertes integriertes Modul `ConsoleProgressBar` mit einer Breite von 100 und dem Zeichen „=“:

```

key:
[                                                                ] 0% 0/1048576 B
key:
[=====] 100% 1048576/1048576 B

```

Passen Sie es an, indem Sie das einfache Fortschrittsbalkenformat verwenden

Im folgenden Beispiel wird der standardmäßige Fortschrittsbalken mithilfe der integrierten Anzeige `PlainProgressBarFormat` [<add link>](#) mit einem benutzerdefinierten Zeichen und einer benutzerdefinierten Breite angepasst. Die `PlainProgressBarFormat` Displays sind nur schwarz. Die Farbe ändert sich nicht je nach Übertragungsstatus.

Das Beispiel ändert sich:

- das Zeichen von der Standardeinstellung „#“ zu „*“
- Balkenbreite von der Standardeinstellung 50 bis 25
- die progressBarFormat Standardinstanz von ColoredTransferProgressBarFormat bis PlainProgressBarFormat

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\PlainProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

$progressBar = new ConsoleProgressBar(
    progressBarChar: '*',
    progressBarWidth: 25,
    progressBarFormat: new PlainProgressBarFormat(),
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file.txt";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();
print_r($result);
```

Die folgende Abbildung zeigt die Konsolenausgabe für ein benutzerdefiniertes integriertes System `ConsoleProgressBar` mit einem einfachen Fortschrittsbalkenformat mit einer Breite von 25 und dem Zeichen „*“:

```
key:
[                               ] 0%

key:
[*****                         ] 44%

key:
[*****] 100%
```

Passen Sie es an, indem Sie ein neues Balkenformat erstellen

Im folgenden Beispiel wird der standardmäßige Fortschrittsbalken angepasst, indem ein neues Balkenformat (Erweiterung `ProgressBarFormat` [<add link>](#)) verwendet wird, anstatt ein integriertes Format anzupassen.

```
<?php

use Aws\S3\S3Client;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Progress\ConsoleProgressBar;
use Aws\S3\S3Transfer\Progress\ProgressBarFormat;
use Aws\S3\S3Transfer\Progress\SingleProgressTracker;
use Aws\S3\S3Transfer\S3TransferManager;

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

// Implement a custom bar format.
class PercentBarFormat extends ProgressBarFormat
{
    public const FORMAT_TEMPLATE = "|object_name| - |percent|% [|transferred| |unit|
ytes]";
    public const FORMAT_PARAMETERS = [
        'object_name',      // `uploadRequestArgs` Key parameter.
        'percent',          // Percent transferred.
        // 'progress_bar',   Optional, if used, default is a `ConsoleProgressBar`.
        // 'to_be_transferred', Optional.
        'transferred',      // Default is bytes transferred.
    ];
}
```



```

        'unit',                // Default is 'B'.
    ];

    public function getFormatTemplate(): string
    {
        return self::FORMAT_TEMPLATE;
    }

    public function getFormatParameters(): array
    {
        return self::FORMAT_PARAMETERS;
    }

    protected function getFormatDefaultParameterValues(): array
    {
        return [];
    }
}

// Create an instance of the custom implementation
$progressBarFormat = new PercentBarFormat();
$progressBar = new ConsoleProgressBar(
    progressBarFormat: $progressBarFormat,
);
$progressTracker = new SingleProgressTracker(
    progressBar: $progressBar,
);
$s3Client = new S3Client([
    'region' => 'us-east-2',
]);
$s3TransferManager = new S3TransferManager($s3Client);
$source = "/path/file";

$result = $s3TransferManager->upload(
    new UploadRequest(
        source: $source,
        uploadRequestArgs: [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'key',
        ],
        progressTracker: $progressTracker,
    )
)->wait();

```

```
print_r($result);
```

Die folgende Abbildung zeigt die Konsolenausgabe für ein benutzerdefiniertes integriertes System `ConsoleProgressBar` mit einem benutzerdefinierten Fortschrittsbalkenformat:

```
key - 0% [0 Bytes]
key - 11% [1153434 Bytes]
key - 55% [5767168 Bytes]
key - 100% [10485760 Bytes]
```

Ein benutzerdefiniertes Balkenformat akzeptiert mehrere Parameter:

- `object_name`
- Prozent
- `progress_bar`
- `to_be_transferiert` werden
- übertragen
- Einheit

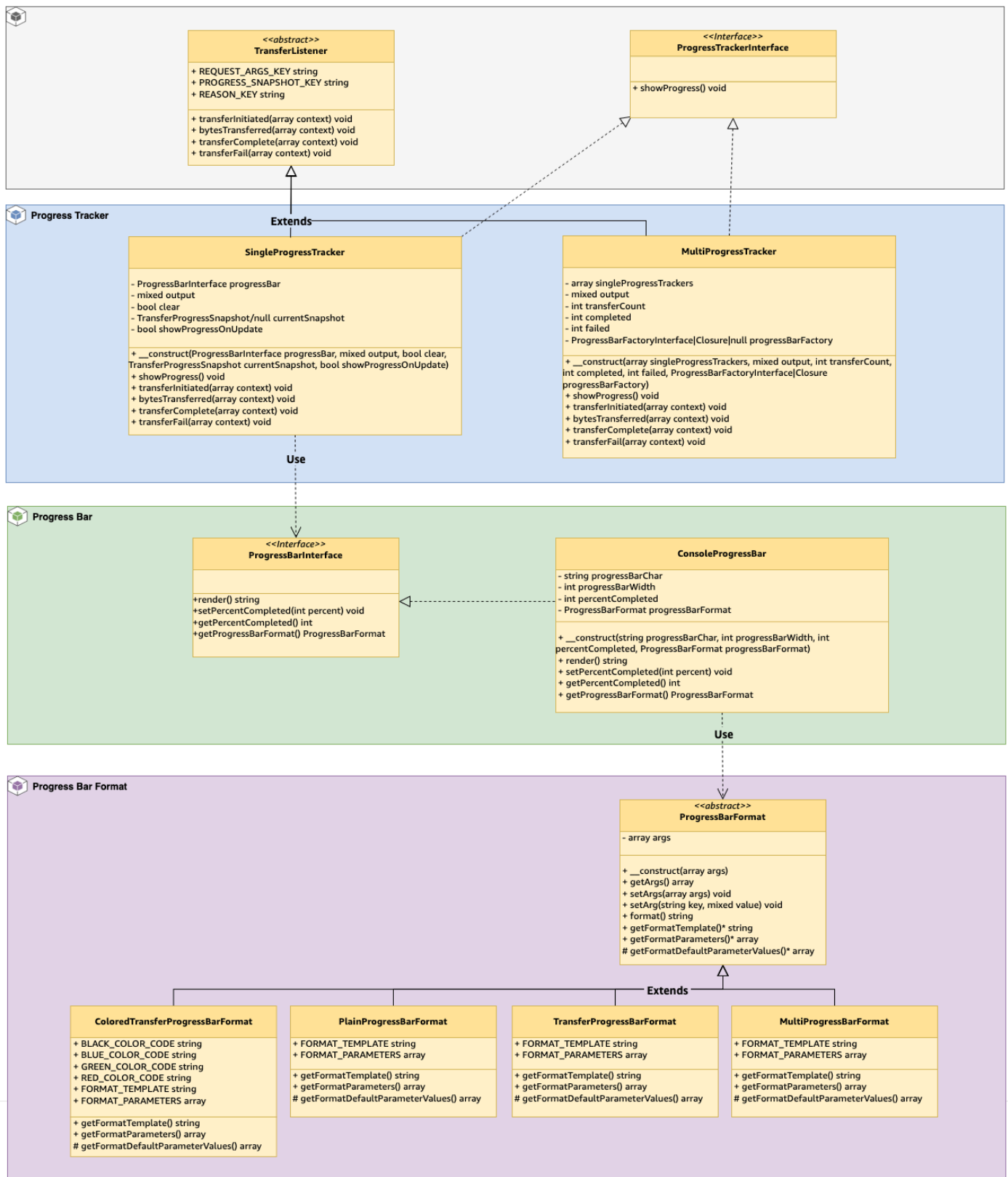
Sie können die durch Trennzeichen (Pipe-Zeichen) getrennten Parameternamen kombinieren, um die Ausgabe anzupassen. In dem Beispiel wird die folgende Vorlage verwendet. Die `getFormatTemplate` Methode der `PercentBarFormat` Klasse gibt diese Vorlage zurück:

```
"|object_name| - |percent|% [|transferred| |unit|ytes]"
```

Klassendiagramm: Tracking-Komponenten von S3 Transfer Manager

Das folgende Klassendiagramm zeigt, wie die Tracking-Komponenten im S3 Transfer Manager zusammenarbeiten. Wenn Sie diese Zusammenhänge verstehen, können Sie [benutzerdefinierte Listener und die Fortschrittsverfolgung](#) implementieren.

S3 Transfer Manager tracking components



Fehlerbehandlung

S3 Transfer Manager verwendet [Promises](#) für asynchrone Operationen. Sie können Fehler mit der `then` Methode `otherwise` oder des Versprechens behandeln. Sie können Ihre Implementierung auch in einen `try-catch` Block packen.

Promise Fehlerbehandlung

Verwenden von `otherwise`:

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'file.txt',
        ]
    )
);

$uploadPromise->otherwise(function (Throwable $reason) {
    echo "Upload failed: " . $reason->getMessage() . "\n";
});
```

Verwenden von `then`:

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\Models\UploadResult;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
```

```
$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'file.txt',
        ]
    )
);

$uploadPromise->then(
    function (UploadResult $result) {
        echo "Upload succeeded!\n";
    },
    function (Throwable $error) {
        echo "Upload failed: " . $error->getMessage() . "\n";
    }
)->wait();
```

Try-Catch-Block verwenden

```
<?php

use Aws\S3\S3Transfer\Exception\S3TransferException;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/file.txt',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'file.txt',
        ]
    )
);

try {
```

```

    $uploadPromise->wait();
} catch (S3TransferException $exception) {
    echo "Upload failed: " . $exception->getMessage() . "\n";
}

```

<Add more details about S3TransferManager exception and also a link to the API reference for it.>

Richtlinien für den Ausfall von Verzeichnisvorgängen

Für Verzeichnisoperationen können Sie auch einen Rückruf für Fehlerrichtlinien angeben. Dieser Callback behandelt Fehler bei einzelnen Dateien und entscheidet, ob der Verzeichnisupload fortgesetzt werden soll.

Example der Verwendung einer 'failure_policy'-Funktion für den Betrieb **uploadDirectory**

```

<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $uploadDirectoryRequestArgs,
                $reason,
                $uploadDirectoryResponse
            ) {
                echo "Failed to upload {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}\n";
                echo "So far, uploaded: " .
                    "{$uploadDirectoryResponse->getObjectsUploaded()}, " .
                    "failed: {$uploadDirectoryResponse->getObjectsFailed()}\n";
            }
        ]
    )
);

```

```

        // Return true to continue with other files,
        // or throw an exception to abort.
        return true;
    },
]
)
);
$uploadDirPromise->wait();

```

Example der Verwendung einer 'failure_policy'-Funktion für den Betrieb **downloadDirectory**

```

<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Log errors but continue with other files.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/directory',
        [],
        [
            'failure_policy' => function (
                $requestArgs,
                $downloadDirectoryRequestArgs,
                $reason,
                $downloadDirectoryResponse
            ) {
                error_log("Failed to download {$requestArgs['Key']}: " .
                    "{$reason->getMessage()}");

                // If we've had too many failures, abort the entire operation.
                if ($downloadDirectoryResponse->getObjectsFailed() > 10) {
                    throw new \Exception(
                        "Too many download failures, aborting operation"
                    );
                }
            }
        ]
    )
);

```

```
        // Return void to continue with other files.
        return;
    },
    ]
)
);
$downloadDirPromise->wait();
```

Fortgeschrittene Verwendung

In diesem Abschnitt werden erweiterte Nutzungsmuster und Techniken für den S3 Transfer Manager behandelt.

Mehrteilige Uploads

S3 Transfer Manager verwendet automatisch mehrteilige Uploads für große Dateien. Sie können dieses Verhalten mit Konfigurationsoptionen anpassen.

```
<?php

use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$uploadPromise = $transferManager->upload(
    new UploadRequest(
        '/path/to/large/file.mp4',
        [
            'Bucket' => 'amzn-s3-demo-bucket',
            'Key'     => 'videos/large-file.mp4',
        ],
        [
            // Use multipart upload for files larger than 100MB.
            'multipart_upload_threshold_bytes' => 100 * 1024 * 1024,

            // Use 25MB parts for multipart uploads.
            'target_part_size_bytes'          => 25 * 1024 * 1024,
        ]
    )
);
```



```
$uploadPromise->wait();
```

Mehrteilige Downloads

Sie können das Verhalten mehrteiliger Downloads anpassen.

```
<?php

use Aws\S3\S3Transfer\AbstractMultipartDownloader;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

$downloadPromise = $transferManager->download(
    new DownloadRequest(
        's3://amzn-s3-demo-bucket/large-file.mp4',
        [],
        [
            // Use 25MB parts for multipart downloads.
            'target_part_size_bytes' => 25 * 1024 * 1024,

            // Use ranged-based download instead of part-based.
            'multipart_download_type' =>
                AbstractMultipartDownloader::RANGED_GET_MULTIPART_DOWNLOADER,
        ]
    )
);

$downloadPromise->wait();
```

Weitere Informationen zu anderen Mitgliedern der `AbstractMultipartDownloader` Klasse finden Sie in der API-Dokumentation [*<add link>*](#).

Benutzerdefinierte Filter

Sie können Filterfunktionen verwenden, um Dateien für Verzeichnisoperationen selektiv hoch- oder herunterzuladen.

Verzeichnisfilter hochladen

Weitere Informationen zu Parameterinformationen finden Sie in der [aufrufbaren Filter-Option](#) der `uploadDirectory` Methode.

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Upload files modified in the last 24 hours.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
            'filter' => function ($file) {
                $modTime = filemtime($file);
                return (time() - $modTime) < 86400; // 24 hours
            },
        ]
    )
);
$uploadDirPromise->wait();
```

Laden Sie den Verzeichnisfilter herunter

Weitere Informationen zu Parameterinformationen finden Sie in der [aufrufbaren Filter-Option](#) der `downloadDirectory` Methode.

```
<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);
```

```
// Download files with a specific prefix.
$downloadDirPromise = $transferManager->downloadDirectory(
    new DownloadDirectoryRequest(
        'amzn-s3-demo-bucket',
        '/path/to/directory',
        [],
        [
            'filter' => function ($key) {
                return strpos($key, 'reports/2023/') === 0;
            },
        ]
    )
);
$downloadDirPromise->wait();
```

Rückrufe vor der Anforderung

Sie können die Anforderungsparameter für jede Datei mithilfe von Pre-Request-Callbacks ändern. Weitere Informationen zu Parametern finden Sie im Folgenden:

- [put_object_request_callback](#), aufrufbare Option der Methode `uploadDirectory`
- [download_object_request_modifier](#), [aufrufbare Option](#) der Methode `downloadDirectory`

Example eines Rückrufs für die Methode **uploadDirectory** vor der Anfrage

```
<?php

use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\S3TransferManager;

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

$transferManager = new S3TransferManager(null, []);

// Set custom metadata for each uploaded file.
$uploadDirPromise = $transferManager->uploadDirectory(
    new UploadDirectoryRequest(
        '/path/to/directory',
        'amzn-s3-demo-bucket',
        [],
        [
```

```

        'upload_object_request_modifier' => function ($args) {
            $extension = pathinfo($args['Key'], PATHINFO_EXTENSION);

            switch ($extension) {
                case 'jpg':
                case 'jpeg':
                    $args['ContentType'] = 'image/jpeg';
                    break;
                case 'png':
                    $args['ContentType'] = 'image/png';
                    break;
                case 'pdf':
                    $args['ContentType'] = 'application/pdf';
                    break;
            }

            $args['Metadata'] = [
                'uploaded_at' => date('Y-m-d H:i:s'),
            ];
        },
    ],
)
);
$uploadDirPromise->wait();

```

Ready-to-use Beispiele

Dieser Abschnitt enthält eine vollständige Hilfsklasse mit statischen Methoden für allgemeine S3 Transfer Manager-Operationen. Die Helper-Klasse vereinfacht die Verwendung. Sie kümmert sich intern um Initialisierung und Konfiguration.

S3-Klasse TransferHelper

```

<?php

use Aws\S3\S3Transfer\Models\DownloadDirectoryRequest;
use Aws\S3\S3Transfer\Models\DownloadFileRequest;
use Aws\S3\S3Transfer\Models\DownloadRequest;
use Aws\S3\S3Transfer\Models\UploadDirectoryRequest;
use Aws\S3\S3Transfer\Models\UploadRequest;
use Aws\S3\S3Transfer\S3TransferManager;
use Psr\Http\Message\StreamInterface;

```

```
class S3TransferHelper
{
    /**
     * Upload a file from a local path to S3.
     */
    public static function uploadFile(
        string $filePath,
        string $bucket,
        string $key
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $filePath,
            [
                'Bucket' => $bucket,
                'Key'     => $key,
            ],
            []
        );

        return $transferManager->upload($uploadRequest)->wait()->getResult();
    }

    /**
     * Upload from a stream to S3
     */
    public static function uploadFromStream(
        StreamInterface $stream,
        string           $bucket,
        string           $key,
        bool             $trackProgress = true,
    ): array
    {
        $transferManager = new S3TransferManager();

        $uploadRequest = new UploadRequest(
            $stream,
            [
                'Bucket' => $bucket,
                'Key'     => $key,
            ],
            ['track_progress' => $trackProgress]
        );
    }
}
```

```

    );

    return $transferManager->upload($uploadRequest)->wait()->getResult();
}

/**
 * Download a file from S3 to a local path
 */
public static function downloadFile(
    string $bucket,
    string $key,
    string $destinationPath,
    bool   $failsWhenDestinationExists = false,
    bool   $trackProgress = true,
): void
{
    $transferManager = new S3TransferManager();
    $downloadRequest = new DownloadRequest(
        [
            'Bucket' => $bucket,
            'Key'     => $key
        ],
        [],
        ['track_progress' => $trackProgress]
    );

    $downloadFileRequest = new DownloadFileRequest(
        $destinationPath,
        $failsWhenDestinationExists,
        $downloadRequest
    );

    $transferManager->downloadFile($downloadFileRequest)->wait();
}

/**
 * Upload an entire directory to S3
 */
public static function uploadDirectory(
    string $sourceDirectory,
    string $bucket,
    string $s3Prefix = '',
    bool   $trackProgress = true,
): array

```

```

{
    $transferManager = new S3TransferManager();
    $uploadDirectoryRequest = new UploadDirectoryRequest(
        $sourceDirectory,
        $bucket,
        [],
        [
            's3_prefix'      => $s3Prefix,
            'track_progress' => $trackProgress,
        ]
    );

    $result = $transferManager->uploadDirectory($uploadDirectoryRequest)->wait();

    return [
        'uploaded' => $result->getObjectsUploaded(),
        'failed'   => $result->getObjectsFailed(),
    ];
}

/**
 * Download directory from S3
 */
public static function downloadDirectory(
    string $bucket,
    string $destinationDirectory,
    string $s3Prefix = '',
    bool   $trackProgress = true,
): array
{
    $transferManager = new S3TransferManager();
    $downloadDirectoryRequest = new DownloadDirectoryRequest(
        $bucket,
        $destinationDirectory,
        [],
        [
            's3_prefix'      => $s3Prefix,
            'track_progress' => $trackProgress,
        ]
    );

    $result = $transferManager->downloadDirectory($downloadDirectoryRequest)->wait();
}

```

```
        return [
            'downloaded' => $result->getObjectsDownloaded(),
            'failed'      => $result->getObjectsFailed(),
        ];
    }
}
```

Beispiele zur Verwendung der Helper-Klasse

Die folgenden Beispiele zeigen, wie die `S3TransferHelper` Klasse verwendet wird.

```
<?php

require __DIR__ . '/../vendor/autoload.php';
require __DIR__ . '/S3TransferHelper.php';

// Upload a local file
$result = S3TransferHelper::uploadFile(
    '/path/to/local/document.pdf',
    'amzn-s3-demo-bucket',
    'documents/document.pdf'
);

echo "File uploaded successfully. ETag: " . $result['ETag'] . "\n";

// Download a file to local path
S3TransferHelper::downloadFile(
    'amzn-s3-demo-bucket',
    'documents/document.pdf',
    '/path/to/local/downloaded-document.pdf',
    false, // Don't fail if destination exists
    true  // Track progress
);

echo "File downloaded successfully!\n";

// Upload entire directory
$result = S3TransferHelper::uploadDirectory(
    '/path/to/local/photos',
    'amzn-s3-demo-bucket',
    'photos/vacation-2023/', // S3 prefix
    true // Track progress
);
```



```
echo "Directory upload completed.\n";
echo "Uploaded: {$result['uploaded']} files\n";
echo "Failed: {$result['failed']} files\n";

// Download directory from S3
$result = S3TransferHelper::downloadDirectory(
    'amzn-s3-demo-bucket',
    '/path/to/local/downloads',
    'photos/vacation-2023/', // S3 prefix to download
    true // Track progress
);

echo "Directory download completed.\n";
echo "Downloaded: {$result['downloaded']} files\n";
echo "Failed: {$result['failed']} files\n";
```

Übertragen Sie Verzeichnisse mit der AWS SDK für PHP Version 3 zu und von Amazon S3

Sie verwenden die Transfer Klasse in AWS SDK für PHP Version 3, um ganze Verzeichnisse in einen Amazon S3 S3-Bucket hochzuladen und ganze Buckets in ein lokales Verzeichnis herunterzuladen.

Ein lokales Verzeichnis auf Amazon S3 hochladen

Das [Aws\S3\Transfer](#) Objekt führt die Übertragungen durch. Das folgende Beispiel zeigt, wie ein lokales Dateiverzeichnis rekursiv in einen Amazon S3 S3-Bucket hochgeladen wird.

```
// Create an S3 client.
$client = new \Aws\S3\S3Client([
    'region' => 'us-west-2',
    'version' => '2006-03-01',
]);

// Where the files will be sourced from.
$source = '/path/to/source/files';

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

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

// Perform the transfer synchronously.
```

```
$directoryTransfer->transfer();
```

In diesem Beispiel haben wir einen Amazon S3 S3-Client erstellt, ein Transfer Objekt erstellt und die Übertragung synchron durchgeführt.

Das vorherige Beispiel zeigt die Mindestmenge an Code, die für die Durchführung einer Übertragung erforderlich ist. Ein Transfer Objekt kann Übertragungen auch asynchron durchführen und verfügt über verschiedene Konfigurationsoptionen, mit denen Sie die Übertragungen anpassen können.

Sie können die lokalen Dateien in einen „Unterordner“ eines Amazon S3 S3-Buckets hochladen, indem Sie in der `s3://` URI ein key prefix angeben. Im folgenden Beispiel werden die lokalen Dateien auf dem Datenträger in den bucket-Bucket hochgeladen und die Dateien unter dem Schlüsselpräfix `foo` gespeichert.

```
$source = '/path/to/source/files';  
$dest = 's3://bucket/foo';  
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);  
$directoryTransfer->transfer();
```

Einen Amazon S3 S3-Bucket herunterladen

Sie können einen Amazon S3 S3-Bucket rekursiv in ein lokales Verzeichnis auf der Festplatte herunterladen, indem Sie das `$source` Argument als Amazon S3 S3-URI (z. B. `s3://bucket`) und das `$dest` Argument als Pfad zu einem lokalen Verzeichnis angeben.

```
// Where the files will be sourced from.  
$source = 's3://bucket';  
  
// Where the files will be transferred to.  
$dest = '/path/to/destination/dir';  
  
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);  
$directoryTransfer->transfer();
```

Note

Das SDK erstellt automatisch alle erforderlichen Verzeichnisse, wenn die Objekte im Bucket heruntergeladen werden.

Sie können nach dem Bucket ein key prefix in die Amazon S3 S3-URI aufnehmen, um nur Objekte herunterzuladen, die in einem „Pseudoordner“ gespeichert sind. Das folgende Beispiel lädt nur Dateien herunter, die unter dem Schlüsselpräfix „/ foo“ des betreffenden Buckets gespeichert sind.

```
$source = 's3://bucket/foo';  
$dest = '/path/to/destination/dir';  
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);  
$directoryTransfer->transfer();
```

Konfiguration

Der Transfer-Objektconstructor akzeptiert die folgenden Argumente:

\$client

Das `Aws\ClientInterface`-Objekt für die Ausführung der Übertragungen.

\$source(Zeichenfolge |) **Iterator**

Die zu übertragenden Quelldaten. Dies kann auf einen lokalen Pfad auf der Festplatte (z. B. `/path/to/files`) oder auf einen Amazon S3 S3-Bucket (z. B. `s3://bucket`) verweisen. Die `s3://`-URI kann auch ein Schlüsselpräfix enthalten, das verwendet werden kann, um nur Objekte unter einem gemeinsamen Präfix zu übertragen.

Wenn das `$source` Argument ein Amazon S3 S3-URI ist, muss das `$dest` Argument ein lokales Verzeichnis sein (und umgekehrt).

Neben der Bereitstellung eines Zeichenfolgenwerts können Sie auch ein `\Iterator`-Objekt angeben, das absolute Dateinamen erzeugt. Wenn Sie ein `\Iterator` Objekt angeben, müssen Sie eine `base_dir` Option im `$options` assoziativen Array angeben.

\$dest

Das Ziel, an das die Dateien übertragen werden. Wenn das `$source` Argument ein lokaler Pfad auf der Festplatte ist, `$dest` muss es sich um einen Amazon S3 S3-Bucket-URI handeln (z. B. `s3://bucket`). Wenn das `$source` Argument ein Amazon S3 S3-Bucket-URI ist, muss das `$dest` Argument ein lokaler Pfad auf der Festplatte sein.

\$options

Ein assoziatives Array mit Übertragungsoptionen. Die folgenden Übertragungsoptionen sind gültig:

add_content_md5 (bool)

Wird auf `true` gesetzt, um die MD5 Prüfsumme für Uploads zu berechnen.

base_dir (string)

Basisverzeichnis der Quelle, wenn `$source` ein Iterator ist. Wenn die `$source`-Option kein Array ist, wird diese Option ignoriert.

before (aufrufbar)

Ein Callback, der vor jeder Übertragung aufgerufen wird. Der Callback sollte eine Funktionssignatur wie `function (Aws\Command $command) {...}` haben. Der bereitgestellte Befehl ist `GetObjectPutObject`, `CreateMultipartUpload`, `UploadPart` oder `CompleteMultipartUpload`.

mup_threshold (int)

Größe in Bytes, für die ein mehrteiliger Upload statt `PutObject` verwendet werden soll. Standardwert 16777216 (16 MB).

concurrency (int, default=5)

Anzahl der Dateien, die gleichzeitig hochgeladen werden. Die ideale Nebenläufigkeitswert variiert abhängig von der Anzahl der Dateien, die hochgeladen werden, und der durchschnittlichen Größe der einzelnen Datei. Im Allgemeinen profitieren kleinere Dateien von einer höheren Nebenläufigkeit, während größere Dateien dies nicht tun.

debug (bool)

Auf `true` setzen, um Debugging-Informationen für Übertragungen auszugeben. Auf eine `fopen()`-Ressource setzen, um statt auf `STDOUT` in einen bestimmten Stream zu schreiben.

Asynchrone Übertragungen

Das `Transfer`-Objekt ist eine Instance von `GuzzleHttp\Promise\PromisorInterface`. Das bedeutet, dass die Übertragung asynchron stattfinden kann und durch Aufruf der `promise`-Methode des Objekts ausgelöst wird.

```
$source = '/path/to/source/files';  
$dest = 's3://bucket';  
$directoryTransfer = new \Aws\S3\Transfer($client, $source, $dest);
```

```
// Initiate the transfer and get a promise.
$promise = $directoryTransfer->promise();

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

Das Versprechen wird abgelehnt, wenn eine Datei nicht übertragen werden kann. Sie können die fehlgeschlagene Übertragung asynchron unter Verwendung der `otherwise`-Methode des Promise verarbeiten. Die `otherwise`-Funktion akzeptiert einen Callback, der aufgerufen wird, wenn ein Fehler auftritt. Der Rückruf akzeptiert das `$reason` für die Ablehnung, was normalerweise eine Instanz von `Aws\Exception\AwsException` (obwohl ein beliebiger Wert an den Rückruf übergeben werden kann).

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

Da das `Transfer`-Objekt ein Promise zurückgibt, können diese Übertragungen gleichzeitig mit anderen asynchronen Promises stattfinden.

Anpassen der Verzeichnisübertragung

Indem Sie dem Konstruktor einen Callback hinzufügen, können Sie die Optionen anpassen, die der Konstruktor ausführt. `Transfer`

```
$uploader = new Transfer($s3Client, $source, $dest, [
    'before' => function (\Aws\Command $command) {
        // Commands can vary for multipart uploads, so check which command
        // is being processed.
        if (in_array($command->getName(), ['PutObject', 'CreateMultipartUpload'])) {
            // Set custom cache-control metadata.
            $command['CacheControl'] = 'max-age=3600';
            // Apply a canned ACL.
            $command['ACL'] = strpos($command['Key'], 'CONFIDENTIAL') === false
                ? 'public-read'
                : 'private';
        }
    },
],
```

```
]);
```

Clientseitige Amazon S3 S3-Verschlüsselung in Version 3 AWS SDK für PHP

Mit client-seitiger Verschlüsselung werden Daten direkt in Ihrer Umgebung verschlüsselt und entschlüsselt. Das bedeutet, dass diese Daten vor der Übertragung an Amazon S3 verschlüsselt werden und Sie sich nicht auf einen externen Dienst verlassen müssen, der die Verschlüsselung für Sie übernimmt. Für neue Implementierungen empfehlen wir die Verwendung von `S3EncryptionClientV3` und `S3EncryptionMultipartUploaderV3` über und `S3EncryptionClientV2` und die veralteten `S3EncryptionClient` Optionen `S3EncryptionMultipartUploaderV2` und `S3EncryptionMultipartUploader`. Es wird empfohlen, dass ältere Implementierungen, die immer noch die veralteten Versionen verwenden, versuchen, zu migrieren. `S3EncryptionClientV3` unterstützt weiterhin die Entschlüsselung von Daten, die mit der älteren Version verschlüsselt wurden. `S3EncryptionClient`

Das AWS SDK für PHP implementiert die [Envelope-Verschlüsselung](#) und verwendet [OpenSSL](#) für die Verschlüsselung und Entschlüsselung. Die Implementierung ist mit [anderen, die ihrer Funktionsunterstützung entsprechen SDKs](#), interoperabel. Sie ist ebenfalls kompatibel mit dem [auf Promises basierenden asynchronen Workflow des SDK](#).

Migrationshandbuch

[Für diejenigen, die versuchen, von den veralteten Clients auf die neuen Clients zu migrieren, gibt es hier einen Migrationsleitfaden für die Migration von Version 1 auf Version 2 und einen Leitfaden zur Migration von Version 2 auf Version 3 hier.](#)

Aufstellen

Um mit der clientseitigen Verschlüsselung zu beginnen, benötigen Sie Folgendes:

- [Ein Verschlüsselungsschlüssel AWS KMS](#)
- Einen [S3-Bucket](#)

Bevor Sie einen Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen. Siehe [Anmeldeinformationen für AWS SDK für PHP Version 3](#).

Verschlüsselung

Für das Hochladen eines verschlüsselten Objekts werden zusätzlich zu den Standardparametern vier zusätzliche `PutObject` Parameter `S3EncryptionClientV3` benötigt:

- '@KmsEncryptionContext' ist ein Schlüssel-Wert-Paar, das verwendet werden kann, um Ihrem verschlüsselten Objekt eine zusätzliche Sicherheitsebene hinzuzufügen. Der Verschlüsselungsclient muss denselben Schlüssel übergeben, was er bei einem GET-Aufruf automatisch tut. Wenn kein zusätzlicher Kontext gewünscht wird, übergeben Sie ein leeres Array.
- @CipherOptionssind zusätzliche Konfigurationen für die Verschlüsselung, einschließlich der zu verwendenden Chiffre und der Schlüsselgröße.
- @MaterialsProviderist ein Anbieter, der sich um die Generierung eines Chiffrierschlüssels und eines Initialisierungsvektors sowie um die Verschlüsselung Ihres Chiffrierschlüssels kümmert.
- @CommitmentPolicyist eine Richtlinienoption, die festlegt, wie ein Objekt entweder mit oder ohne Schlüsselzusage gelesen wird und wie ein Objekt mit oder ohne Schlüsselzusage geschrieben wird.

```
use Aws\S3\S3Client;
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\Kms\KmsClient;
use Aws\Crypto\KmsMaterialsProviderV3;

// Let's construct our S3EncryptionClient using an S3Client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$bucket = 'the-bucket-name';
$key = 'the-file-name';
$cipherOptions = [
    'Cipher' => 'gcm',
```

```
'KeySize' => 256,  
// Additional configuration options  
];  
  
$result = $encryptionClient->putObject([  
    '@MaterialsProvider' => $materialsProvider,  
    '@CipherOptions' => $cipherOptions,  
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',  
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],  
    'Bucket' => $bucket,  
    'Key' => $key,  
    'Body' => fopen('file-to-encrypt.txt', 'r'),  
]);
```

Note

Zusätzlich zu den Amazon S3- und AWS KMS basierten Servicefehlern erhalten Sie möglicherweise geworfene `InvalidArgumentException` Objekte, wenn Sie nicht richtig konfiguriert '@CipherOptions' sind.

Entschlüsselung

Beim Herunterladen und Entschlüsseln eines Objekts gibt es zusätzlich zu den Standardparametern fünf zusätzliche `GetObject` Parameter, von denen zwei erforderlich sind. Der Client erkennt die grundlegenden Verschlüsselungsoptionen für Sie.

- **'@SecurityProfile'**: Wenn auf 'V3' gesetzt, werden nur Objekte angezeigt, die in V3-kompatibel verschlüsselt sind

Format kann entschlüsselt werden. Wenn dieser Parameter auf 'V3_AND_LEGACY' gesetzt wird, können auch Objekte entschlüsselt werden, die im V1-kompatiblen Format verschlüsselt wurden. Um die Migration zu unterstützen, setzen Sie @ auf 'V3_AND_LEGACY'.
SecurityProfile Verwenden Sie 'V3' nur für die Entwicklung neuer Anwendungen.

- **'@MaterialsProvider'** ist ein Anbieter, der sich um die Generierung eines Chiffrierschlüssels und eines Initialisierungsvektors kümmert, als

sowie die Verschlüsselung Ihres Chiffrierschlüssels.

- **'@KmsAllowDecryptWithAnyCmk'**: (optional) Wenn Sie diesen Parameter auf true setzen, wird die Entschlüsselung aktiviert

ohne dem Konstruktor von eine KMS-Schlüssel-ID zur Verfügung zu stellen. MaterialsProvider Der Standardwert ist "false".
- **'@CipherOptions'** (optional) sind zusätzliche Konfigurationen für die Verschlüsselung, darunter zu verwendende Chiffre und Schlüsselgröße.
- **@CommitmentPolicy** Richtlinienoption, die festlegt, wie ein Objekt gelesen wird, entweder mit Schlüsselzusage oder ohne Schlüsselzusage und wie ein Objekt mit oder ohne Schlüsselzusage geschrieben wird.

```
$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Note

Zusätzlich zu den Amazon S3- und AWS KMS basierten Servicefehlern erhalten Sie möglicherweise geworfene `InvalidArgumentException` Objekte, wenn Sie nicht richtig konfiguriert '@CipherOptions' sind.

Konfiguration der Chiffre

'Cipher' (string)

Verschlüsselungsmethode, die der Verschlüsselungsclient bei der Verschlüsselung verwendet. Derzeit wird nur 'gcm' unterstützt.

⚠ Important

PHP wurde [in Version 7.1 aktualisiert](#), um zusätzliche Parameter zu unterstützen, die für die [Verschlüsselung](#) und [Entschlüsselung](#) unter Verwendung von OpenSSL für die GCM-Verschlüsselung erforderlich sind. Für PHP-Versionen 7.0 und früher wird ein Polyfill für GCM-Unterstützung bereitgestellt und von den Verschlüsselungsclients und verwendet. `S3EncryptionClientV2` `S3EncryptionMultipartUploaderV2`. Allerdings wird die Leistung bei großen Eingaben mit Polyfill viel langsamer sein als mit der nativen Implementierung für PHP 7.1+. Daher kann es notwendig sein, ältere PHP-Versionsumgebungen zu aktualisieren, um sie effektiv nutzen zu können.

'KeySize' (int)

Die Länge des für die Verschlüsselung zu generierenden Inhaltsverschlüsselungsschlüssels. Der Standardwert ist 256 Bit. Gültige Konfigurationsoptionen sind 256 Bit.

'Aad' (string)

Optionale „Zusätzliche Authentifizierungsdaten“ für Ihre verschlüsselte Nutzlast. Diese Informationen werden bei der Entschlüsselung validiert. Aad ist nur verfügbar, wenn Sie die „gcm“-Verschlüsselung anwenden.

⚠ Important

Zusätzliche Authentifizierungsdaten werden nicht von allen unterstützt, AWS SDKs weshalb andere SDKs möglicherweise nicht in der Lage sind, mit diesem Parameter verschlüsselte Dateien zu entschlüsseln.

Strategien für Metadaten

Sie haben auch die Möglichkeit, eine Instance einer Klasse bereitzustellen, die das `Aws\Crypto\MetadataStrategyInterface` implementiert. Diese einfache Schnittstelle übernimmt das Speichern und Laden des `Aws\Crypto\MetadataEnvelope`, der Ihre Daten für die Envelope-Verschlüsselung enthält. Das SDK bietet zwei Klassen, die Folgendes implementieren: `Aws\S3\Crypto\HeadersMetadataStrategy` und `Aws\S3\Crypto\InstructionFileMetadataStrategy`. Standardmäßig wird `HeadersMetadataStrategy` verwendet.

```

$strategy = new InstructionFileMetadataStrategy(
    $s3Client
);

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

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => false,
    '@MaterialsProvider' => $materialsProvider,
    '@SecurityProfile' => 'V3',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => $strategy,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);

```

Klassennamenkonstanten für `HeadersMetadataStrategy` und `InstructionFileMetadataStrategy` können durch Aufruf von `::class` bereitgestellt werden.

```

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

```

 Note

Wenn nach dem Hochladen einer Anweisungsdatei ein Fehler auftritt, wird diese nicht automatisch gelöscht.

Mehrteilige Uploads

Mehrteilige Upload mit clientseitiger Verschlüsselung sind ebenfalls möglich. Der `Aws\S3\Crypto\S3EncryptionMultipartUploaderV3` bereitet den Quellstream vor dem Hochladen für die Verschlüsselung vor. Das Erstellen funktioniert ähnlich wie mit dem `Aws\S3\MultipartUploader` und dem `Aws\S3\Crypto\S3EncryptionClientV3`. Der `S3EncryptionMultipartUploaderV3` kann dieselbe '@MetadataStrategy'-Option wie der `S3EncryptionClientV3` verarbeiten, ebenso wie alle verfügbaren '@CipherOptions'-Konfigurationen.

```
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    $kmsKeyId
);

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

$multipartUploader = new S3EncryptionMultipartUploaderV3(
    new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'profile' => 'default',
    ]),
    fopen('large-file-to-encrypt.txt', 'r'),
```

```
[
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    'bucket' => $bucket,
    'key' => $key,
]
);
$multipartUploader->upload();
```

Note

Zusätzlich zu den Amazon S3- und AWS KMS basierten Servicefehlern erhalten Sie möglicherweise geworfene `InvalidArgumentException` Objekte, wenn Sie nicht richtig konfiguriert '@CipherOptions' sind.

Schutz der Datenintegrität mit Prüfsummen

Amazon Simple Storage Service (Amazon S3) bietet die Möglichkeit, beim Hochladen eines Objekts eine Prüfsumme anzugeben. Wenn Sie eine Prüfsumme angeben, wird diese zusammen mit dem Objekt gespeichert und kann beim Herunterladen des Objekts überprüft werden.

Prüfsummen bieten eine zusätzliche Ebene der Datenintegrität bei der Übertragung von Dateien. Mit Prüfsummen können Sie die Datenkonsistenz überprüfen, indem Sie sicherstellen, dass die empfangene Datei mit der Originaldatei übereinstimmt. Weitere Informationen zu Prüfsummen mit Amazon S3 finden Sie im [Amazon Simple Storage Service-Benutzerhandbuch](#), einschließlich der [unterstützten Algorithmen](#).

Sie haben die Flexibilität, den Algorithmus auszuwählen, der Ihren Anforderungen am besten entspricht, und das SDK die Prüfsumme berechnen zu lassen. Alternativ können Sie mithilfe eines der unterstützten Algorithmen einen vorab berechneten Prüfsummenwert angeben.

Note

Ab Version 3.337.0 von bietet das SDK standardmäßigen Integritätsschutz AWS SDK für PHP, indem es automatisch eine Prüfsumme für Uploads berechnet. CRC32 Das SDK berechnet diese Prüfsumme, wenn Sie keinen vorab berechneten Prüfsummenwert angeben oder wenn Sie keinen Algorithmus angeben, den das SDK zur Berechnung einer Prüfsumme verwenden soll.

[Das SDK bietet auch globale Einstellungen für den Schutz der Datenintegrität, die Sie extern festlegen können. Weitere Informationen finden Sie im Referenzhandbuch und im Tools-Referenzhandbuch.AWS SDKs](#)

 Important

Um mit dem CRC32C Algorithmus arbeiten zu können, erfordert Ihre PHP-Umgebung die [Installation der AWS Common Runtime \(AWS CRT\)](#) -Erweiterung.

Wir behandeln Prüfsummen in zwei Anforderungsphasen: beim Hochladen eines Objekts und beim Herunterladen eines Objekts.

Hochladen eines Objekts

Sie laden Objekte auf Amazon S3 hoch, indem Sie die [PutObject-Methode](#) von `S3Client` verwenden. Verwenden Sie das `ChecksumAlgorithm` Paar im Parameter-Array, um die Prüfsummenberechnung zu aktivieren und den Algorithmus anzugeben.

```
$client = new \Aws\S3\S3Client(['region' => 'us-east-2']); // See the note below.
$result = $client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'key',
    'ChecksumAlgorithm' => 'CRC32',
    'Body' => 'Object contents to test the checksum.'
]);
```

Wenn Sie mit der Anfrage keinen Prüfsummenalgorithmus angeben, variiert das Prüfsummenverhalten je nach der Version des verwendeten SDK, wie in der folgenden Tabelle dargestellt.

Prüfsummenverhalten, wenn kein Prüfsummenalgorithmus bereitgestellt wird

PHP-SDK-Version	Verhalten der Prüfsumme
früher als 3.337.0	Das SDK berechnet nicht automatisch eine CRC-basierte Prüfsumme und gibt sie in der Anfrage an.

PHP-SDK-Version	Verhalten der Prüfsumme
3.337.0 oder höher	Das SDK verwendet den CRC32 Algorithmus zur Berechnung der Prüfsumme und stellt sie in der Anfrage bereit. Amazon S3 validiert die Integrität der Übertragung, indem es seine eigene CRC32 Prüfsumme berechnet und sie mit der vom SDK bereitgestellten Prüfsumme vergleicht. Wenn die Prüfsummen übereinstimmen, wird die Prüfsumme zusammen mit dem Objekt gespeichert.

Verwenden Sie einen vorberechneten Prüfsummenwert

Ein mit der Anfrage bereitgestellter vorberechneter Prüfsummenwert deaktiviert die automatische Berechnung durch das SDK und verwendet stattdessen den angegebenen Wert.

Das folgende Beispiel zeigt eine Anfrage mit einer vorberechneten Prüfsumme. SHA256

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

$client = new S3Client([
    'region' => 'us-east-1',
]);

// Calculate the SHA256 checksum of the contents to be uploaded.
$contents = 'Object contents to test the checksum.';
$body = Psr7\Utils::streamFor($contents);
$sha256 = base64_encode(Psr7\Utils::hash($body, 'sha256', true));

$result = $client->putObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'key',
    'Body' => $body,
    'ChecksumSHA256' => $sha256
]);
```

Wenn Amazon S3 feststellt, dass der Prüfsummenwert für den angegebenen Algorithmus falsch ist, gibt der Service eine Fehlerantwort zurück.

Mehrteilige Uploads

Sie können Prüfsummen auch bei mehrteiligen Uploads verwenden.

Wie im folgenden Beispiel gezeigt, geben Sie den Prüfsummenalgorithmus als Schlüssel-Wert-Paar im Array des Konstruktors an. `paramsMultipartUploader`

```
$s3Client = new S3Client([
    'region' => 'us-east-1'
]);

$stream = fopen("/path/to/large/file", "r");

$mpUploader = new MultipartUploader($s3Client, $stream, [
    'bucket' => 'amzn-s3-demo-bucket',
    'key' => 'key',
    'params' => ['ChecksumAlgorithm' => 'CRC32']
]);
```

Herunterladen eines Objekts

Wenn Sie die [GetObject](#) `GetObject` `getObject(input: )` verwenden, um ein Objekt herunterzuladen, validiert das SDK automatisch die Prüfsumme, wenn der Wert des Schlüssels lautet. `ChecksumMode enabled`

Die Anfrage im folgenden Codeausschnitt weist das SDK an, die Prüfsumme in der Antwort zu validieren, indem es die Prüfsumme berechnet und die Werte vergleicht.

```
$result = $client->getObject([
    'Bucket' => 'amzn-s3-demo-bucket',
    'Key' => 'test-checksum-key',
    'ChecksumMode' => 'enabled',
]);
```

Note

Wenn das Objekt nicht mit einer Prüfsumme hochgeladen wurde, findet keine Überprüfung statt.

Codebeispiele mit Anleitungen für die AWS SDK für PHP

Dieser Abschnitt enthält Codebeispiele, die allgemeine AWS Szenarien demonstrieren, in denen der verwendet wird AWS SDK für PHP.

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Themen

- [CloudFront Amazon-Beispiele mit der AWS SDK für PHP Version 3](#)
- [Signieren von benutzerdefinierten CloudSearch Amazon-Domainanfragen mit AWS SDK für PHP Version 3](#)
- [CloudWatch Amazon-Beispiele mit der AWS SDK für PHP Version 3](#)
- [EC2 Amazon-Beispiele mit der AWS SDK für PHP Version 3](#)
- [Signieren einer Amazon OpenSearch Service-Suchanfrage mit AWS SDK für PHP Version 3](#)
- [AWS Identity and Access Management Beispiele mit der AWS SDK für PHP Version 3](#)
- [AWS Key Management Service Beispiele mit der AWS SDK für PHP Version 3](#)
- [Amazon Kinesis Kinesis-Beispiele mit AWS SDK für PHP Version 3](#)
- [AWS Elemental MediaConvert Beispiele mit der AWS SDK für PHP Version 3](#)
- [Amazon S3 S3-Beispiele mit AWS SDK für PHP Version 3](#)
- [Verwaltung von Geheimnissen mit der Secrets Manager API und der AWS SDK für PHP Version 3](#)
- [Amazon SES SES-Beispiele mit AWS SDK für PHP Version 3](#)
- [Amazon SNS SNS-Beispiele mit AWS SDK für PHP Version 3](#)
- [Amazon SQS SQS-Beispiele mit AWS SDK für PHP Version 3](#)
- [Senden Sie Ereignisse mit der AWS SDK für PHP Version 3 an EventBridge globale Amazon-Endpunkte](#)

CloudFront Amazon-Beispiele mit der AWS SDK für PHP Version 3

Amazon CloudFront ist ein AWS Webservice, der die Bereitstellung statischer und dynamischer Webinhalte von Ihrem eigenen Webserver oder einem AWS Server wie Amazon S3 beschleunigt. CloudFront stellt Inhalte über ein weltweites Netzwerk von Rechenzentren bereit, die als Edge-Standorte bezeichnet werden. Wenn ein Benutzer Inhalte anfordert, mit denen Sie sie verteilen CloudFront, wird er an den Edge-Standort weitergeleitet, der die niedrigste Latenz bietet. Falls sich der Inhalt dort noch nicht im Cache befindet, ruft CloudFront eine Kopie vom Ursprungs-Server ab, stellt sie bereit, und speichert sie dann für zukünftige Anfragen im Cache.

Weitere Informationen CloudFront dazu finden Sie im [Amazon CloudFront Developer Guide](#).

Der gesamte Beispielcode für die AWS SDK für PHP Version 3 ist [hier verfügbar GitHub](#).

Verwaltung von CloudFront Amazon-Distributionen mithilfe der CloudFront API und der AWS SDK für PHP Version 3

Amazon CloudFront speichert Inhalte an weltweiten Edge-Standorten im Cache, um die Verteilung statischer und dynamischer Dateien zu beschleunigen, die Sie auf Ihrem eigenen Server oder auf einem Amazon-Dienst wie Amazon S3 und Amazon EC2 speichern. Wenn Benutzer Inhalte von Ihrer Website anfordern, werden sie CloudFront vom nächstgelegenen Edge-Standort aus bereitgestellt, sofern die Datei dort zwischengespeichert ist. Andernfalls wird eine Kopie der Datei CloudFront abgerufen, bereitgestellt und dann für die nächste Anfrage zwischengespeichert. Das Speichern von Inhalten im Cache eines Edge-Standorts reduziert die Latenz von ähnlichen Benutzeranfragen in dieser Region.

Für jede CloudFront Verteilung, die Sie erstellen, geben Sie an, wo sich der Inhalt befindet und wie er verteilt werden soll, wenn Benutzer Anfragen stellen. Dieses Thema konzentriert sich auf Verteilungen von statischen und dynamischen Dateien wie HTML-, CSS-, JSON- und Bilddateien. Informationen zur Verwendung CloudFront mit Video-on-Demand finden Sie unter [On-Demand-Video und Live-Streaming mit CloudFront](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie eine Distribution mit [CreateDistribution](#).
- Holen Sie sich eine Distribution mit [GetDistribution](#).
- Listet Distributionen auf mit [ListDistributions](#).
- Aktualisieren Sie Distributionen mit. [UpdateDistributions](#)
- Deaktivieren Sie Distributionen mit. [DisableDistribution](#)

- Löschen Sie Distributionen mit. [DeleteDistributions](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von Amazon CloudFront finden Sie im [Amazon CloudFront Developer Guide](#).

Erstellen Sie eine CloudFront Distribution

Erstellen Sie eine Distribution aus einem Amazon S3 S3-Bucket. Im folgenden Beispiel werden optionale Parameter auskommentiert, aber Standardwerte werden angezeigt. Zum Hinzufügen von Anpassungen zu Ihrer Verteilung kommentieren Sie sowohl den Wert als auch den Parameter in `$distribution` aus.

Verwenden Sie den [CreateDistribution](#) Vorgang, um eine CloudFront Verteilung zu erstellen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
function createS3Distribution($cloudFrontClient, $distribution)
{
    try {
        $result = $cloudFrontClient->createDistribution([
            'DistributionConfig' => $distribution
        ]);

        $message = '';

        if (isset($result['Distribution']['Id'])) {
            $message = 'Distribution created with the ID of ' .

```

```

        $result['Distribution']['Id'];
    }

    $message .= ' and an effective URI of ' .
        $result['@metadata']['effectiveUri'] . '.';

    return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e['message'];
}
}

function createsTheS3Distribution()
{
    $originName = 'my-unique-origin-name';
    $s3BucketURL = 'amzn-s3-demo-bucket.s3.amazonaws.com';
    $callerReference = 'my-unique-caller-reference';
    $comment = 'my-comment-about-this-distribution';
    $defaultCacheBehavior = [
        'AllowedMethods' => [
            'CachedMethods' => [
                'Items' => ['HEAD', 'GET'],
                'Quantity' => 2
            ],
            'Items' => ['HEAD', 'GET'],
            'Quantity' => 2
        ],
        'Compress' => false,
        'DefaultTTL' => 0,
        'FieldLevelEncryptionId' => '',
        'ForwardedValues' => [
            'Cookies' => [
                'Forward' => 'none'
            ],
            'Headers' => [
                'Quantity' => 0
            ],
            'QueryString' => false,
            'QueryStringCacheKeys' => [
                'Quantity' => 0
            ]
        ],
        'LambdaFunctionAssociations' => ['Quantity' => 0],
        'MaxTTL' => 0,
    ];
}

```

```

        'MinTTL' => 0,
        'SmoothStreaming' => false,
        'TargetOriginId' => $originName,
        'TrustedSigners' => [
            'Enabled' => false,
            'Quantity' => 0
        ],
        'ViewerProtocolPolicy' => 'allow-all'
    ];
    $enabled = false;
    $origin = [
        'Items' => [
            [
                'DomainName' => $s3BucketURL,
                'Id' => $originName,
                'OriginPath' => '',
                'CustomHeaders' => ['Quantity' => 0],
                'S3OriginConfig' => ['OriginAccessIdentity' => '']
            ]
        ],
        'Quantity' => 1
    ];
    $distribution = [
        'CallerReference' => $callerReference,
        'Comment' => $comment,
        'DefaultCacheBehavior' => $defaultCacheBehavior,
        'Enabled' => $enabled,
        'Origins' => $origin
    ];

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo createS3Distribution($cloudFrontClient, $distribution);
}

// Uncomment the following line to run this code in an AWS account.
// createsTheS3Distribution();

```

Rufen Sie eine CloudFront Verteilung ab

Verwenden Sie den [GetDistribution](#) Vorgang, um den Status und die Details einer bestimmten CloudFront Verteilung abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
function getDistribution($cloudFrontClient, $distributionId)
{
    try {
        $result = $cloudFrontClient->getDistribution([
            'Id' => $distributionId
        ]);

        $message = '';

        if (isset($result['Distribution']['Status'])) {
            $message = 'The status of the distribution with the ID of ' .
                $result['Distribution']['Id'] . ' is currently ' .
                $result['Distribution']['Status'];
        }

        if (isset($result['@metadata']['effectiveUri'])) {
            $message .= ', and the effective URI is ' .
                $result['@metadata']['effectiveUri'] . '.';
        } else {
            $message = 'Error: Could not get the specified distribution. ' .
                'The distribution\'s status is not available.';
        }

        return $message;
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function getsADistribution()
{
    $distributionId = 'E1BTGP2EXAMPLE';

    $cloudFrontClient = new Aws\CloudFront\CloudFrontClient([
        'profile' => 'default',
        'version' => '2018-06-18',
        'region' => 'us-east-1'
    ]);

    echo getDistribution($cloudFrontClient, $distributionId);
}

// Uncomment the following line to run this code in an AWS account.
// getsADistribution();
```

CloudFront Verteilungen auflisten

Rufen Sie mithilfe dieses Vorgangs eine Liste der vorhandenen CloudFront Distributionen in der angegebenen AWS Region von Ihrem Girokonto ab. [ListDistributions](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
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();
```

Aktualisieren Sie eine Distribution CloudFront

Das Aktualisieren einer CloudFront Distribution ähnelt dem Erstellen einer Distribution. Bei der Aktualisierung einer Verteilung sind jedoch mehr Felder erforderlich und alle Werte müssen berücksichtigt werden. Zur Vornahme von Änderungen an einer vorhandenen Verteilung empfehlen wir, zuerst die vorhandene Verteilung abzurufen und die zu aktualisierenden Werte im `$distribution`-Array zu ändern.

Verwenden Sie den [UpdateDistribution](#) Vorgang, um eine angegebene CloudFront Distribution zu aktualisieren.

Importe

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

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

Beispiel-Code

```
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();

```

Deaktiviert eine CloudFront Distribution

Zum Deaktivieren oder Entfernen einer Verteilung ändern Sie ihren Status von „Bereitgestellt“ in „Deaktiviert“.

Verwenden Sie den [DisableDistribution](#) Vorgang, um die angegebene CloudFront Distribution zu deaktivieren.

Importe

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;

```

Beispiel-Code

```

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();

```

Löscht eine CloudFront Distribution

Sobald eine Verteilung den Status „Deaktiviert“ aufweist, können Sie sie löschen.

Verwenden Sie den [DeleteDistribution](#) Vorgang, um eine angegebene CloudFront Distribution zu entfernen.

Importe

```

require 'vendor/autoload.php';

use Aws\Exception\AwsException;

```

Beispiel-Code

```

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();
```

Verwaltung von CloudFront Amazon-Invalidierungen mithilfe der CloudFront API und der Version 3 AWS SDK für PHP

Amazon CloudFront speichert Kopien statischer und dynamischer Dateien an weltweiten Edge-Standorten im Cache. Erstellen Sie zum Entfernen oder Aktualisieren einer Datei auf allen Edge-Standorten eine Aufhebung für alle Dateien bzw. für eine Gruppe von Dateien.

Die ersten 1.000 Aufhebungen jedes Kalendermonats sind kostenlos. Weitere Informationen zum Entfernen von Inhalten von einem CloudFront Edge-Standort finden Sie unter Dateien für [ungültig erklären](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie eine Invalidierung der Verteilung mit. [CreateInvalidation](#)
- Rufen Sie eine Distributions-Invalidierung ab mit. [GetInvalidation](#)
- Listet Distributionen auf mit. [ListInvalidations](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon CloudFront finden Sie im [Amazon CloudFront Developer Guide](#).

Erstellen Sie eine Invalidierung der Distribution

Erstellen Sie eine Invalidierung der CloudFront Distribution, indem Sie den Pfad für die Dateien angeben, die Sie entfernen müssen. Dieses Beispiel hebt die Gültigkeit aller Dateien in der Verteilung auf, aber Sie können bestimmte Dateien unter Items identifizieren.

Verwenden Sie den Vorgang, um eine Invalidierung der CloudFront [CreateInvalidation](#) Verteilung zu erstellen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
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();

```

Holen Sie sich eine Distributions-Invalidierung

Verwenden Sie den Vorgang, um den Status und die Details einer CloudFront Verteilung für ungültig zu erklären. [GetInvalidation](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
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();
```

Ungültigerklärungen von Verteilungen auflisten

Verwenden Sie den Vorgang, um alle aktuellen CloudFront Verteilungsungültigkeiten aufzulisten.

[ListInvalidations](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
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();

```

Amazon CloudFront URLs mit AWS SDK für PHP Version 3 signieren

Signiert URLs ermöglichen es Ihnen, Benutzern Zugriff auf Ihre privaten Inhalte zu gewähren. Eine signierte URL enthält zusätzliche Informationen (z. B. Ablaufzeit), mit denen Sie den Zugriff auf Ihre Inhalte besser kontrollieren können. Diese zusätzlichen Informationen sind in einer Richtlinienanweisung enthalten, die entweder auf einer vordefinierten oder einer benutzerdefinierten Richtlinie basieren. Informationen darüber, wie Sie private Distributionen einrichten und warum Sie

unterschreiben müssen URLs, finden Sie unter [Serving Private Content through Amazon CloudFront im Amazon CloudFront Developer Guide](#).

- Erstellen Sie mit [getSignedUrl](#) eine signierte CloudFront Amazon-URL.
- Erstellen Sie ein signiertes CloudFront Amazon-Cookie mit [getSignedCookie](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von Amazon CloudFront finden Sie im [Amazon CloudFront Developer Guide](#).

Signieren CloudFront URLs für private Distributionen

Sie können eine URL mit dem CloudFront Client im SDK signieren. Zuerst müssen Sie ein Objekt `CloudFrontClient` erstellen. Sie können eine CloudFront URL für eine Videoressource entweder mithilfe einer vorgefertigten oder einer benutzerdefinierten Richtlinie signieren.

Importe

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

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

Beispiel-Code

```
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();

```

Verwenden Sie beim Erstellen eine benutzerdefinierte Richtlinie CloudFront URLs

Um eine benutzerdefinierte Richtlinie zu verwenden, geben Sie den Schlüssel `policy` anstelle von `expires` an.

Importe


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

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

Beispiel-Code

```
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();

```

Verwenden Sie eine CloudFront signierte URL

Die Form der signierten URL unterscheidet sich in Abhängigkeit davon, ob die URL, die Sie signieren, das Schema „HTTP“ oder „RTMP“ verwendet. Im Fall von „HTTP“ wird die vollständige, absolute URL zurückgegeben. Für „RTMP“ wird nur die relative URL zurückgegeben. Dies liegt daran, dass einige Player den Host und den Pfad als separate Parameter benötigen.

Das folgende Beispiel zeigt, wie Sie die signierte URL verwenden können, um eine Webseite zu erstellen, auf der ein Video mit angezeigt wird [JWPlayer](#). Dieselbe Technik würde auch für andere Player gelten [FlowPlayer](#), erfordert aber einen anderen clientseitigen Code.

```

<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>
```

Signieren von CloudFront Cookies für private Distributionen

Als Alternative zu signierten URLs können Sie Kunden auch über signierte Cookies Zugriff auf eine private Distribution gewähren. Signierte Cookies ermöglichen Ihnen den Zugriff auf mehrere Dateien mit beschränkten Rechten, z. B. alle Dateien für ein Video im HLS-Format oder alle Dateien im Bereich der Abonnenten einer Website. Weitere Informationen darüber, warum Sie signierte Cookies anstelle von signierten Cookies verwenden möchten URLs (oder umgekehrt), finden Sie unter [Wählen zwischen signierten URLs und signierten Cookies](#) im Amazon CloudFront Developer Guide.

Das Erstellen eines signierten Cookies ähnelt dem Erstellen einer signierten URL. Der einzige Unterschied ist die aufgerufene Methode (`getSignedCookie` statt `getSignedUrl`).

Importe

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

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

Beispiel-Code

```
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();
```

Verwenden Sie beim Erstellen von CloudFront Cookies eine benutzerdefinierte Richtlinie

Wie bei `getSignedUrl` können Sie anstelle eines Parameters `'policy'` einen Parameter `expires` und einen Parameter `url` verwenden, um einen Cookie mit einer benutzerdefinierten Richtlinie zu unterschreiben. Eine benutzerdefinierte Richtlinie kann Platzhalter im Ressourcenschlüssel enthalten. Dadurch können Sie ein einzelnes signiertes Cookie für mehrere Dateien erstellen.

`getSignedCookie` gibt ein Array von Schlüssel-Wert-Paaren zurück, die alle als Cookies festgelegt werden müssen, um den Zugriff auf eine private Distribution zu ermöglichen.

Importe

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

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

Beispiel-Code

```
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://d13149jEXAMPLE.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();
```

Sende CloudFront Cookies an den Guzzle-Client

Sie können diese Cookies auch an eine `GuzzleHttp\Cookie\CookieJar` zur Verwendung mit einem Guzzle-Client übergeben.

```
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');
```

Weitere Informationen finden Sie unter [Verwenden signierter Cookies](#) im Amazon CloudFront Developer Guide.

Signieren von benutzerdefinierten CloudSearch Amazon-Domainanfragen mit AWS SDK für PHP Version 3

CloudSearch Amazon-Domainanfragen können über das hinaus angepasst werden, was von der unterstützt wird AWS SDK für PHP. [In Fällen, in denen Sie benutzerdefinierte Anfragen an Domains stellen müssen, die durch IAM-Authentifizierung geschützt sind, können Sie die Anmeldeinformationsanbieter und Unterzeichner des SDK verwenden, um jede PSR-7-Anfrage zu signieren.](#)

Wenn Sie beispielsweise das Handbuch [Erste Schritte mit Cloud Search](#) verwenden und eine IAM-geschützte Domäne für [Schritt 3](#) verwenden wollen, müssen Sie Ihre Anforderung wie folgt signieren und ausführen.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- [Signieren Sie eine Anfrage mit dem Signaturprotokoll mithilfe von SignatureV4 AWS](#) .

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

CloudSearch Amazon-Domainanfrage signieren

Importe

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

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

Beispiel-Code

```
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 = '7kbitd6nyiglhdmtssxEXAMPLE';
    $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 Amazon-Beispiele mit der AWS SDK für PHP Version 3

Amazon CloudWatch (CloudWatch) ist ein Webservice, der Ihre Amazon Web Services Services-Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit überwacht. Sie können CloudWatch damit Metriken sammeln und verfolgen. Dabei handelt es sich um Variablen, die Sie für Ihre Ressourcen und Anwendungen messen können. CloudWatch Alarme senden Benachrichtigungen oder nehmen auf der Grundlage von von Ihnen festgelegter Regeln automatisch Änderungen an den Ressourcen vor, die Sie überwachen.

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Themen

- [Arbeiten mit CloudWatch Amazon-Alarmen mit AWS SDK für PHP Version 3](#)
- [Metriken von Amazon CloudWatch mit AWS SDK für PHP Version 3 abrufen](#)
- [Veröffentlichen von benutzerdefinierten Metriken in Amazon CloudWatch mit AWS SDK für PHP Version 3](#)
- [Senden von Ereignissen an Amazon CloudWatch Events mit AWS SDK für PHP Version 3](#)
- [Verwenden von Alarmaktionen mit CloudWatch Amazon-Alarmen mit AWS SDK für PHP Version 3](#)

Arbeiten mit CloudWatch Amazon-Alarmen mit AWS SDK für PHP Version 3

Ein CloudWatch Amazon-Alarm überwacht eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, basierend auf dem Wert der Metrik im Vergleich zu einem bestimmten Schwellenwert in einer Reihe von Zeiträumen.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Beschreiben Sie einen Alarm mit [DescribeAlarms](#).
- Erstellen Sie einen Alarm mit [PutMetricAlarm](#).

- Löschen Sie einen Alarm mit [DeleteAlarms](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called "Installation"](#).

Alarme beschreiben

Importe

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

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

Beispiel-Code

```
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();
```

Alarm erstellen

Importe

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

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

Beispiel-Code

```
function putMetricAlarm(
    $cloudWatchClient,
    $cloudWatchRegion,
    $alarmName,
    $namespace,
    $metricName,
    $dimensions,
    $statistic,
    $period,
    $comparison,
    $threshold,
    $evaluationPeriods
) {
    try {
        $result = $cloudWatchClient->putMetricAlarm([
            'AlarmName' => $alarmName,
            'Namespace' => $namespace,
            'MetricName' => $metricName,
            'Dimensions' => $dimensions,
            'Statistic' => $statistic,
            'Period' => $period,
            'ComparisonOperator' => $comparison,
            'Threshold' => $threshold,
            'EvaluationPeriods' => $evaluationPeriods
        ]);

        if (isset($result['@metadata']['effectiveUri'])) {
            if (
                $result['@metadata']['effectiveUri'] ==
                'https://monitoring.' . $cloudWatchRegion . '.amazonaws.com'
            ) {
                return 'Successfully created or updated specified alarm.';
            } else {
                return 'Could not create or update specified alarm.';
            }
        } else {
            return 'Could not create or update specified alarm.';
        }
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}
```

```
function putTheMetricAlarm()
{
    $alarmName = 'my-ec2-resources';
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $statistic = 'Average';
    $period = 300;
    $comparison = 'GreaterThanOrEqualToThreshold';
    $threshold = 1;
    $evaluationPeriods = 1;

    $cloudWatchRegion = 'us-east-1';
    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => $cloudWatchRegion,
        'version' => '2010-08-01'
    ]);

    echo putMetricAlarm(
        $cloudWatchClient,
        $cloudWatchRegion,
        $alarmName,
        $namespace,
        $metricName,
        $dimensions,
```

```

        $statistic,
        $period,
        $comparison,
        $threshold,
        $evaluationPeriods
    );
}

// Uncomment the following line to run this code in an AWS account.
// putTheMetricAlarm();

```

Löschen von Alarmen

Importe

```

require 'vendor/autoload.php';

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

```

Beispiel-Code

```

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();
```

Metriken von Amazon CloudWatch mit AWS SDK für PHP Version 3 abrufen

Metriken sind Daten über die Leistung Ihrer Systeme. Sie können die detaillierte Überwachung einiger Ressourcen, z. B. Ihrer EC2 Amazon-Instances, oder Ihrer eigenen Anwendungsmetriken aktivieren.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Metriken auflisten mit [ListMetrics](#).
- Rufen Sie Alarme für eine Metrik ab mit [DescribeAlarmsForMetric](#).
- Rufen Sie Statistiken für eine angegebene Metrik ab mit [GetMetricStatistics](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Auflisten von Metriken

Importe

```
require 'vendor/autoload.php';  
  
use Aws\CloudWatch\CloudWatchClient;  
use Aws\Exception\AwsException;
```


Beispiel-Code

```
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();

```

Rufen Sie Alarme für eine Metrik ab

Importe

```

require 'vendor/autoload.php';

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

```

Beispiel-Code

```

function describeAlarmsForMetric(
    $cloudWatchClient,
    $metricName,
    $namespace,
    $dimensions
) {
    try {
        $result = $cloudWatchClient->describeAlarmsForMetric([
            'MetricName' => $metricName,

```

```

        'Namespace' => $namespace,
        'Dimensions' => $dimensions
    ]);

    $message = '';

    if (isset($result['@metadata']['effectiveUri'])) {
        $message .= 'At the effective URI of ' .
            $result['@metadata']['effectiveUri'] . ":\n\n";

        if (
            (isset($result['MetricAlarms'])) and
            (count($result['MetricAlarms']) > 0)
        ) {
            $message .= 'Matching alarms for ' . $metricName . ":\n\n";

            foreach ($result['MetricAlarms'] as $alarm) {
                $message .= $alarm['AlarmName'] . "\n";
            }
        } else {
            $message .= 'No matching alarms found for ' . $metricName . '.';
        }
    } else {
        $message .= 'No matching alarms found for ' . $metricName . '.';
    }

    return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
}

function describeTheAlarmsForMetric()
{
    $metricName = 'BucketSizeBytes';
    $namespace = 'AWS/S3';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value' => 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'amzn-s3-demo-bucket'
        ]
    ];
}

```

```
    ]
];

$client = new CloudWatchClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-08-01'
]);

echo describeAlarmsForMetric(
    $client,
    $metricName,
    $namespace,
    $dimensions
);
}

// Uncomment the following line to run this code in an AWS account.
// describeTheAlarmsForMetric();
```

Abrufen von Metrikstatistiken

Importe

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

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

Beispiel-Code

```
function getMetricStatistics(
    $client,
    $namespace,
    $metricName,
    $dimensions,
    $startTime,
    $endTime,
    $period,
    $statistics,
    $unit
) {
```

```
try {
    $result = $cloudWatchClient->getMetricStatistics([
        'Namespace' => $namespace,
        'MetricName' => $metricName,
        'Dimensions' => $dimensions,
        'StartTime' => $startTime,
        'EndTime' => $endTime,
        'Period' => $period,
        'Statistics' => $statistics,
        'Unit' => $unit
    ]);

    $message = '';

    if (isset($result['@metadata']['effectiveUri'])) {
        $message .= 'For the effective URI at ' .
            $result['@metadata']['effectiveUri'] . "\n\n";

        if (
            (isset($result['Datapoints'])) and
            (count($result['Datapoints']) > 0)
        ) {
            $message .= "Datapoints found:\n\n";

            foreach ($result['Datapoints'] as $datapoint) {
                foreach ($datapoint as $key => $value) {
                    $message .= $key . ' = ' . $value . "\n";
                }

                $message .= "\n";
            }
        } else {
            $message .= 'No datapoints found.';
        }
    } else {
        $message .= 'No datapoints found.';
    }

    return $message;
} catch (AwsException $e) {
    return 'Error: ' . $e->getAwsErrorMessage();
}
```

```
function getTheMetricStatistics()
{
    // Average number of Amazon EC2 vCPUs every 5 minutes within
    // the past 3 hours.
    $namespace = 'AWS/Usage';
    $metricName = 'ResourceCount';
    $dimensions = [
        [
            'Name' => 'Service',
            'Value' => 'EC2'
        ],
        [
            'Name' => 'Resource',
            'Value' => 'vCPU'
        ],
        [
            'Name' => 'Type',
            'Value' => 'Resource'
        ],
        [
            'Name' => 'Class',
            'Value' => 'Standard/OnDemand'
        ]
    ];
    $startTime = strtotime('-3 hours');
    $endTime = strtotime('now');
    $period = 300; // Seconds. (5 minutes = 300 seconds.)
    $statistics = ['Average'];
    $unit = 'None';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics(
        $cloudWatchClient,
        $namespace,
        $metricName,
        $dimensions,
        $startTime,
        $endTime,
        $period,
```

```
        $statistics,
        $unit
    );

    // Another example: average number of bytes of standard storage in the
    // specified Amazon S3 bucket each day for the past 3 days.

    /*
    $namespace = 'AWS/S3';
    $metricName = 'BucketSizeBytes';
    $dimensions = [
        [
            'Name' => 'StorageType',
            'Value'=> 'StandardStorage'
        ],
        [
            'Name' => 'BucketName',
            'Value' => 'amzn-s3-demo-bucket'
        ]
    ];
    $startTime = strtotime('-3 days');
    $endTime = strtotime('now');
    $period = 86400; // Seconds. (1 day = 86400 seconds.)
    $statistics = array('Average');
    $unit = 'Bytes';

    $cloudWatchClient = new CloudWatchClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-08-01'
    ]);

    echo getMetricStatistics($cloudWatchClient, $namespace, $metricName,
    $dimensions, $startTime, $endTime, $period, $statistics, $unit);
    */
}

// Uncomment the following line to run this code in an AWS account.
// getTheMetricStatistics();
```

Veröffentlichen von benutzerdefinierten Metriken in Amazon CloudWatch mit AWS SDK für PHP Version 3

Metriken sind Daten über die Leistung Ihrer Systeme. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen festgelegten Zeitraum. Er führt eine oder mehrere Aktionen durch, die vom Wert der Metrik im Vergleich zu einem Schwellenwert in einer Reihe von Zeiträumen abhängt.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Veröffentlichen Sie metrische Daten mit [PutMetricData](#).
- Erstellen Sie einen Alarm mit [PutMetricAlarm](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Veröffentlichen Sie metrische Daten

Importe

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

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

Beispiel-Code

```
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();
```

Alarm erstellen

Importe

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

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

Beispiel-Code

```
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();
```

Senden von Ereignissen an Amazon CloudWatch Events mit AWS SDK für PHP

Version 3

CloudWatch Events liefert nahezu in Echtzeit einen Stream von Systemereignissen, die Änderungen an den Ressourcen von Amazon Web Services (AWS) für verschiedene Ziele beschreiben. Mit einfachen Regeln können Sie Ereignisse zuordnen und sie zu einer oder mehreren Zielfunktionen oder Streams umleiten.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie eine Regel mit [PutRule](#).
- Fügen Sie einer Regel Ziele hinzu mit [PutTargets](#).
- Senden Sie benutzerdefinierte Ereignisse an CloudWatch Ereignisse mithilfe von [PutEvents](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Erstellen einer Regel

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

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

Fügen Sie Ziele zu einer Regel hinzu

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

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

Benutzerdefinierte Ereignisse senden

Importe

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

Beispiel-Code

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

Verwenden von Alarmaktionen mit CloudWatch Amazon-Alarmen mit AWS SDK für PHP Version 3

Verwenden Sie Alarmaktionen, um Alarme zu erstellen, die Ihre EC2 Amazon-Instances automatisch stoppen, beenden, neu starten oder wiederherstellen. Sie können die Aktionen zum Anhalten oder Beenden nutzen, wenn eine Instance nicht mehr ausgeführt werden muss. Sie können die Aktionen zum Neustarten oder Wiederherstellen verwenden, um diese Instances automatisch neu zu starten.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Aktivieren Sie Aktionen für bestimmte Alarme mithilfe von [EnableAlarmActions](#).
- Deaktivieren Sie Aktionen für bestimmte Alarme mit [DisableAlarmActions](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called "Installation"](#).

Aktivieren von Alarmaktionen

Importe

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

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

Beispiel-Code

```
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();

```

Deaktivieren von Alarmaktionen

Importe

```

require 'vendor/autoload.php';

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

```

Beispiel-Code

```

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 Amazon-Beispiele mit der AWS SDK für PHP Version 3

Amazon Elastic Compute Cloud (Amazon EC2) ist ein Webservice, der virtuelles Server-Hosting in der Cloud bietet. Er wurde entwickelt, um Entwicklern Cloud-Computing im Web-Scale-Maßstab zu erleichtern, indem er eine skalierbare Rechenkapazität bietet.

[Der gesamte Beispielcode für AWS SDK für PHP ist hier verfügbar. GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Themen

- [Verwaltung von EC2 Amazon-Instances mit AWS SDK für PHP Version 3](#)
- [Elastic IP-Adressen mit Amazon EC2 mit AWS SDK für PHP Version 3 verwenden](#)
- [Verwenden von Regionen und Verfügbarkeitszonen für Amazon EC2 mit AWS SDK für PHP Version 3](#)
- [Arbeiten mit EC2 Amazon-Schlüsselpaaren mit AWS SDK für PHP Version 3](#)
- [Arbeiten mit Sicherheitsgruppen in Amazon EC2 mit AWS SDK für PHP Version 3](#)

Verwaltung von EC2 Amazon-Instances mit AWS SDK für PHP Version 3

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Beschreiben Sie EC2 Amazon-Instances mithilfe von [DescribeInstances](#).
- Aktivieren Sie die detaillierte Überwachung für eine laufende Instance mithilfe von [MonitorInstances](#).
- Deaktivieren Sie die Überwachung für eine laufende Instanz mit [UnmonitorInstances](#).
- Starten Sie ein Amazon EBS-backed AMI, das Sie zuvor nicht mehr verwenden. [StartInstances](#)
- Stoppen Sie die Verwendung einer Amazon EBS-gestützten Instance. [StopInstances](#)
- Fordern Sie einen Neustart einer oder mehrerer Instances an mit. [RebootInstances](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Beschreiben von Instances

Importe

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

use Aws\Ec2\Ec2Client;
```

Beispiel-Code

```
$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";
    }
}
```

Aktivieren und deaktivieren Sie die Überwachung

Importe

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

Beispiel-Code

```
$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);
```

Starten und beenden Sie eine Instanz

Importe

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

Beispiel-Code

```
$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);
```

Neustarten einer Instance

Importe

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

Beispiel-Code

```
$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);
```

Elastic IP-Adressen mit Amazon EC2 mit AWS SDK für PHP Version 3 verwenden

Eine Elastic IP-Adresse ist eine statische IP-Adresse, die für dynamisches Cloud Computing konzipiert ist. Eine Elastic IP-Adresse ist mit Ihrer verknüpft AWS-Konto. Es ist eine öffentliche IP-Adresse, die aus dem Internet erreichbar ist. Wenn Ihre Instance keine öffentliche IP-Adresse hat, können Sie eine Elastic IP-Adresse mit der Instance verwenden, damit diese mit dem Internet kommunizieren kann.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Beschreiben Sie eine oder mehrere Ihrer Instances mithilfe von [DescribeInstances](#).
- Erwerben Sie eine Elastic IP-Adresse mit [AllocateAddress](#).
- Ordnen Sie eine Elastic IP-Adresse einer Instance zu, indem Sie [AssociateAddress](#).

- Geben Sie eine Elastic IP-Adresse frei mit [ReleaseAddress](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Beschreiben Sie eine Instanz

Importe

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

use Aws\Ec2\Ec2Client;
```

Beispiel-Code

```
$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";
    }
}
```

Ordnen Sie eine Adresse zu und ordnen Sie sie zu

Importe

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

Beispiel-Code

```
$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);
```

Geben Sie eine Adresse frei

Importe

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

Beispiel-Code

```
$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);
```

Verwenden von Regionen und Verfügbarkeitszonen für Amazon EC2 mit AWS SDK für PHP Version 3

Amazon EC2 wird an mehreren Standorten weltweit gehostet. Diese Standorte bestehen aus AWS Regionen und Verfügbarkeitszonen. Jede Region ist ein separates geografisches Gebiet mit mehreren isolierten Standorten, die als Availability Zones bezeichnet werden. Amazon EC2 bietet die Möglichkeit, Instanzen und Daten an mehreren Standorten zu platzieren.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Beschreiben Sie die Availability Zones, die Ihnen zur Verfügung stehen [DescribeAvailabilityZones](#).
- Beschreiben Sie die AWS Regionen, die Ihnen derzeit zur Verfügung stehen [DescribeRegions](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Beschreiben von Availability Zones

Importe

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

Beispiel-Code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeAvailabilityZones();

var_dump($result);
```

Beschreiben von Regionen

Importe

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

Beispiel-Code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeRegions();

var_dump($result);
```

Arbeiten mit EC2 Amazon-Schlüsselpaaren mit AWS SDK für PHP Version 3

Amazon EC2 verwendet Public-Key-Kryptografie, um Anmeldeinformationen zu verschlüsseln und zu entschlüsseln. Bei der Kryptografie für öffentliche Schlüssel werden Daten mithilfe eines

öffentlichen Schlüssels verschlüsselt. Anschließend verwendet der Empfänger den privaten Schlüssel zum Entschlüsseln der Daten. Der öffentliche und der private Schlüssel werden als Schlüsselpaar bezeichnet.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie ein 2048-Bit-RSA-Schlüsselpaar mit [CreateKeyPair](#)
- Löscht ein angegebenes key pair mit [DeleteKeyPair](#).
- Beschreiben Sie eines oder mehrere Ihrer Schlüsselpaare mithilfe von [DescribeKeyPairs](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Erstellen eines Schlüsselpaares

Importe

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

Beispiel-Code

```
$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);
```

Löschen eines Schlüsselpaars

Importe

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

Beispiel-Code

```
$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);
```

Beschreiben von Schlüsselpaaren

Importe

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

Beispiel-Code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeKeyPairs();

var_dump($result);
```

Arbeiten mit Sicherheitsgruppen in Amazon EC2 mit AWS SDK für PHP Version 3

Eine EC2 Amazon-Sicherheitsgruppe fungiert als virtuelle Firewall, die den Datenverkehr für eine oder mehrere Instances steuert. Sie fügen jeder Sicherheitsgruppe Regeln hinzu, um den Datenaustausch mit den verknüpften Instances zu gestatten. Sie können die Regeln für eine Sicherheitsgruppe jederzeit ändern. Die neuen Regeln gelten automatisch für alle Instances, die der Sicherheitsgruppe zugewiesen sind.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Beschreiben Sie eine oder mehrere Ihrer Sicherheitsgruppen mithilfe von [DescribeSecurityGroups](#).
- Fügen Sie einer Sicherheitsgruppe eine Eingangsregel hinzu mit [AuthorizeSecurityGroupIngress](#).
- Erstellen Sie eine Sicherheitsgruppe mit [CreateSecurityGroup](#).
- Löschen Sie eine Sicherheitsgruppe mit [DeleteSecurityGroup](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called "Installation"](#).

Beschreiben von Sicherheitsgruppen

Importe

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

Beispiel-Code

```
$ec2Client = new Aws\Ec2\Ec2Client([
    'region' => 'us-west-2',
    'version' => '2016-11-15',
    'profile' => 'default'
]);

$result = $ec2Client->describeSecurityGroups();

var_dump($result);
```

Fügen Sie eine Eingangsregel hinzu

Importe

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

Beispiel-Code

```
$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);
```

Eine Sicherheitsgruppe erstellen

Importe

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

Beispiel-Code

```
$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";
```

Löschen einer Sicherheitsgruppe

Importe

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

Beispiel-Code

```
$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);
```

Signieren einer Amazon OpenSearch Service-Suchanfrage mit AWS SDK für PHP Version 3

Amazon OpenSearch Service ist ein verwalteter Service, der die Bereitstellung, den Betrieb und die Skalierung von Amazon OpenSearch Service, einer beliebten Open-Source-Such- und Analyse-Engine, vereinfacht. OpenSearch Service bietet direkten Zugriff auf die Amazon OpenSearch Service API. Das bedeutet, dass Entwickler die Tools verwenden können, mit denen sie vertraut sind, sowie robuste Sicherheitsoptionen. Viele Amazon OpenSearch Service-Clients unterstützen das Signieren von Anfragen. Wenn Sie jedoch einen Client verwenden, der dies nicht tut, können Sie beliebige PSR-7-Anfragen mit den integrierten Anmeldeinformationsanbietern und Unterzeichnern von signieren. AWS SDK für PHP

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- [Signieren Sie eine Anfrage mit dem Signaturprotokoll mithilfe von AWS SignatureV4.](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar. GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Eine OpenSearch Serviceanfrage signieren

OpenSearch Der Dienst verwendet [Signature Version 4](#). Das bedeutet, dass Sie Anfragen anhand des Signaturnamens des Dienstes (es in diesem Fall) und der AWS Region Ihrer OpenSearch Service-Domain signieren müssen. Eine vollständige Liste der vom OpenSearch Service unterstützten Regionen finden Sie [auf der Seite AWS Regionen und Endpunkte](#) im Allgemeine

Amazon Web Services-Referenz. In diesem Beispiel signieren wir jedoch Anfragen für eine OpenSearch Service-Domain in der us-west-2 Region.

Sie müssen Anmeldeinformationen angeben, was Sie entweder mit der Standardanbieterkette des SDK oder mit einer beliebigen Form von Anmeldeinformationen tun können, [die unter Anmeldeinformationen für AWS SDK für PHP Version 3](#) beschrieben sind. Sie benötigen außerdem eine [PSR-7-Anfrage](#) (im folgenden Code als `$psr7Request` bezeichnet).

```
// 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 Beispiele mit der AWS SDK für PHP Version 3

AWS Identity and Access Management (IAM) ist ein Webservice, der es Kunden von Amazon Web Services ermöglicht, Benutzer und Benutzerberechtigungen in AWS zu verwalten. Der Service richtet sich an Unternehmen mit mehreren Benutzern oder Systemen in der Cloud, die AWS Produkte verwenden. Mit IAM können Sie Benutzer, Sicherheitsanmeldeinformationen wie Zugriffsschlüssel und Berechtigungen, mit denen gesteuert wird, auf welche AWS Ressourcen Benutzer zugreifen können, zentral verwalten.

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Themen

- [Verwaltung von IAM-Zugriffsschlüsseln mit AWS SDK für PHP Version 3](#)
- [Verwaltung von IAM-Benutzern mit AWS SDK für PHP Version 3](#)

- [Verwenden von IAM-Kontoaliesen mit Version 3 AWS SDK für PHP](#)
- [Arbeiten mit IAM-Richtlinien mit AWS SDK für PHP Version 3](#)
- [Arbeiten mit IAM-Serverzertifikaten mit AWS SDK für PHP Version 3](#)

Verwaltung von IAM-Zugriffsschlüsseln mit AWS SDK für PHP Version 3

Benutzer benötigen ihre eigenen Zugriffstasten, um programmatische Aufrufe tätigen zu können AWS. Um diesen Bedarf zu decken, können Sie Zugriffsschlüssel (Zugriffsschlüssel IDs und geheime Zugriffsschlüssel) für IAM-Benutzer erstellen, ändern, anzeigen oder rotieren. Wenn Sie einen Zugriffsschlüssel erstellen, lautet der Status standardmäßig Aktiv. Dies bedeutet, dass der Benutzer den Zugriffsschlüssel für API-Aufrufe verwenden kann.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen geheimen Zugriffsschlüssel und die entsprechende Zugriffsschlüssel-ID mithilfe von [CreateAccessKey](#).
- Gibt Informationen über den Zugriffsschlüssel zurück, der einem IAM-Benutzer IDs zugeordnet ist, der [ListAccessKeys](#)
- Rufen Sie Informationen darüber ab, wann ein Zugriffsschlüssel zuletzt verwendet wurde mit [GetAccessKeyLastUsed](#).
- Ändern Sie den Status eines Zugriffsschlüssels von Aktiv in Inaktiv oder umgekehrt, indem Sie [UpdateAccessKey](#).
- Löschen Sie ein Zugriffsschlüsselpaar, das einem IAM-Benutzer zugeordnet ist, mithilfe von [DeleteAccessKey](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Erstellen eines Zugriffsschlüssels

Importe

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

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

Beispiel-Code

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

Auflisten von Zugriffsschlüsseln

Importe

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

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

Beispiel-Code

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

Ruft Informationen über die letzte Verwendung eines Zugriffsschlüssels ab

Importe

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

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

Beispiel-Code

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

Aktualisieren eines Zugriffsschlüssels

Importe

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

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

Beispiel-Code

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

Löschen eines Zugriffsschlüssels

Importe

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

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

Beispiel-Code

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

Verwaltung von IAM-Benutzern mit AWS SDK für PHP Version 3

Ein IAM-Benutzer ist eine Entität, die Sie erstellen, AWS um die Person oder den Dienst darzustellen, mit AWS dem er interagiert. Ein Benutzer AWS besteht aus einem Namen und Anmeldeinformationen.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen neuen IAM-Benutzer mit [CreateUser](#).
- Listet IAM-Benutzer auf, die. [ListUsers](#)
- Aktualisieren Sie einen IAM-Benutzer mit. [UpdateUser](#)
- Rufen Sie Informationen über einen IAM-Benutzer ab, indem Sie. [GetUser](#)
- Löschen Sie einen IAM-Benutzer mit. [DeleteUser](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Erstellen eines IAM-Benutzers

Importe

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

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

Beispiel-Code

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

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

IAM-Benutzer auflisten

Importe

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

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

Beispiel-Code

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

Aktualisieren Sie einen IAM-Benutzer

Importe

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

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

Beispiel-Code

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

Holen Sie sich Informationen über einen IAM-Benutzer

Importe

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

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

Beispiel-Code

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

Löschen Sie einen IAM-Benutzer

Importe

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

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

Beispiel-Code

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

Verwenden von IAM-Kontoaliasen mit Version 3 AWS SDK für PHP

Wenn Sie möchten, dass die URL für Ihre Anmeldeseite Ihren Firmennamen oder eine andere benutzerfreundliche Kennung anstelle Ihrer AWS-Konto ID enthält, können Sie einen Alias für Ihre AWS-Konto ID erstellen. Wenn Sie einen AWS-Konto Alias erstellen, ändert sich die URL Ihrer Anmeldeseite, sodass der Alias enthalten ist.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Alias mit [CreateAccountAlias](#).
- Listet den Alias auf, der der AWS-Konto Verwendung zugeordnet ist [ListAccountAliases](#).
- Löschen Sie einen Alias mit [DeleteAccountAlias](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called "Installation"](#).

Erstellen eines Alias

Importe

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

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

Beispiel-Code

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

Auflisten von Konto-Aliasnamen

Importe

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

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

Beispiel-Code

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

Löschen Sie einen Alias

Importe

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

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

Beispiel-Code

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

Arbeiten mit IAM-Richtlinien mit AWS SDK für PHP Version 3

Sie erteilen einem Benutzer Berechtigungen, indem Sie eine Richtlinie erstellen. Eine Richtlinie ist ein Dokument, in dem die Aktionen aufgeführt sind, die ein Benutzer ausführen kann, sowie die Ressourcen, auf die sich diese Aktionen auswirken können. Standardmäßig werden alle Aktionen oder Ressourcen, die nicht explizit erlaubt sind, verweigert. Richtlinien können erstellt und an Benutzer, Benutzergruppen, von Benutzern übernommene Rollen und Ressourcen angehängt werden.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie eine verwaltete Richtlinie mit [CreatePolicy](#).
- Fügen Sie einer Rolle eine Richtlinie hinzu mithilfe von [AttachRolePolicy](#).
- Hängen Sie eine Richtlinie an einen Benutzer an, indem Sie [AttachUserPolicy](#).
- Hängen Sie eine Richtlinie an eine Gruppe an mithilfe von [AttachGroupPolicy](#).
- Entfernen Sie eine Rollenrichtlinie mithilfe von [DetachRolePolicy](#).
- Entfernen Sie eine Benutzerrichtlinie mit [DetachUserPolicy](#).
- Entfernen Sie eine Gruppenrichtlinie mit [DetachGroupPolicy](#).
- Löschen Sie eine verwaltete Richtlinie mit [DeletePolicy](#).
- Löschen Sie eine Rollenrichtlinie mit [DeleteRolePolicy](#).

- Löschen Sie eine Benutzerrichtlinie mit [DeleteUserPolicy](#).
- Löschen Sie eine Gruppenrichtlinie mit [DeleteGroupPolicy](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Erstellen einer Richtlinie

Importe

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

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

Beispiel-Code

```
$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());
}

```

Anfügen einer Richtlinie an eine Rolle

Importe

```

require 'vendor/autoload.php';

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

```

Beispiel-Code

```

$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());
}
```

Anfügen einer Richtlinie an einen Benutzer

Importe

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

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

Beispiel-Code


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

Hängen Sie eine Richtlinie an eine Gruppe an

Importe

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

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

Beispiel-Code

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

Trennen Sie eine Benutzerrichtlinie

Importe

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

Beispiel-Code

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

Trennen Sie eine Gruppenrichtlinie

Importe

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

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

Beispiel-Code

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

Löschen Sie eine Richtlinie

Importe

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

Beispiel-Code

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

Löschen Sie eine Rollenrichtlinie

Importe

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

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

Beispiel-Code

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

Löschen Sie eine Benutzerrichtlinie

Importe

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

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

Beispiel-Code

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

Löschen einer Gruppenrichtlinie

Importe

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

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

Beispiel-Code

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

Arbeiten mit IAM-Serverzertifikaten mit AWS SDK für PHP Version 3

Um HTTPS-Verbindungen zu Ihrer Website oder Anwendung zu aktivieren AWS, benötigen Sie ein SSL/TLS Serverzertifikat. Um ein Zertifikat zu verwenden, das Sie von einem externen Anbieter für Ihre Website oder Anwendung bezogen haben AWS, müssen Sie das Zertifikat in AWS Certificate Manager IAM hochladen oder in IAM importieren.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Listet die in IAM gespeicherten Zertifikate auf unter [ListServerCertificates](#)
- Rufen Sie Informationen zu einem Zertifikat ab mit [GetServerCertificate](#).
- Aktualisieren Sie ein Zertifikat mit [UpdateServerCertificate](#).
- Löschen Sie ein Zertifikat mit [DeleteServerCertificate](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Auflisten von Serverzertifikaten

Importe

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

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

Beispiel-Code

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

Rufen Sie ein Serverzertifikat ab

Importe

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

Beispiel-Code

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

Aktualisieren eines Serverzertifikats

Importe

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

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

Beispiel-Code

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

Löschen eines Serverzertifikats

Importe

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

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

Beispiel-Code

```
$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 Beispiele mit der AWS SDK für PHP Version 3

AWS Key Management Service (AWS KMS) ist ein verwalteter Dienst, der es Ihnen leicht macht, die zur Verschlüsselung Ihrer Daten verwendeten Verschlüsselungsschlüssel zu erstellen und zu kontrollieren. Weitere Informationen zu AWS KMS finden Sie in der [Amazon KMS-Dokumentation](#). Ganz gleich, ob Sie sichere PHP-Anwendungen schreiben oder Daten an andere AWS Dienste senden, es AWS KMS hilft Ihnen, die Kontrolle darüber zu behalten, wer Ihre Schlüssel verwenden und Zugriff auf Ihre verschlüsselten Daten erhalten kann.

Der gesamte Beispielcode für die AWS SDK für PHP Version 3 ist [hier verfügbar GitHub](#).

Themen

- [Arbeiten mit Schlüsseln mithilfe der AWS KMS API und der AWS SDK für PHP Version 3](#)
- [Verschlüsselung und Entschlüsselung von AWS KMS Datenschlüsseln mit der Version 3 AWS SDK für PHP](#)
- [Mit der AWS SDK für PHP Version 3 mit AWS KMS wichtigen Richtlinien arbeiten](#)
- [Arbeiten mit Zuschüssen mithilfe der AWS KMS API und der AWS SDK für PHP Version 3](#)
- [Arbeiten mit Aliasen unter Verwendung der AWS KMS API und der AWS SDK für PHP Version 3](#)

Arbeiten mit Schlüsseln mithilfe der AWS KMS API und der AWS SDK für PHP Version 3

Die primären Ressourcen in AWS Key Management Service () sind [AWS KMS keys](#). Sie können einen KMS-Schlüssel verwenden, um Ihre Daten zu verschlüsseln.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Kunden-KMS-Schlüssel mit [CreateKey](#).
- Generieren Sie einen Datenschlüssel mit [GenerateDataKey](#).
- Zeigen Sie einen KMS-Schlüssel an mit [DescribeKey](#).
- Schlüssel IDs und Schlüssel-ARNs von KMS-Schlüsseln abrufen mithilfe [ListKeys](#) von.
- Aktivieren Sie KMS-Schlüssel mit [EnableKey](#).
- Deaktivieren Sie KMS-Schlüssel mit [DisableKey](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Weitere Informationen zur Verwendung von AWS Key Management Service (AWS KMS) finden Sie im [AWS KMS Entwicklerhandbuch](#).

Erstellen Sie einen KMS-Schlüssel

Verwenden Sie den [CreateKey](#)Vorgang, um einen [KMS-Schlüssel](#) zu erstellen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Erzeugen eines Datenschlüssels

Verwenden Sie den [GenerateDataKey](#)Vorgang, um einen Datenverschlüsselungsschlüssel zu generieren. Diese Operation gibt eine Klartextkopie und eine verschlüsselte Kopie des von ihr erstellten Datenschlüssels zurück. Geben Sie den an, AWS KMS key unter dem der Datenschlüssel generiert werden soll.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Einen KMS-Schlüssel anzeigen

Verwenden Sie den [DescribeKey](#) Vorgang, um detaillierte Informationen zu einem KMS-Schlüssel zu erhalten, einschließlich des Amazon-Ressourcennamens (ARN) und des [Schlüsselstatus](#) des KMS-Schlüssels.

Mit `DescribeKey` können keine Aliasnamen abgerufen werden. Verwenden Sie den [ListAliases](#) Vorgang, um Aliase abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Ruft die Schlüssel-ID und den Schlüssel ARNs eines KMS-Schlüssels ab

Verwenden Sie den [ListAliases](#) Vorgang, um die ID und den ARN des KMS-Schlüssels abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Aktivieren Sie einen KMS-Schlüssel

Verwenden Sie den [EnableKey](#) Vorgang, um einen deaktivierten KMS-Schlüssel zu aktivieren.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Deaktivieren Sie einen KMS-Schlüssel

Verwenden Sie den [DisableKey](#) Vorgang, um einen KMS-Schlüssel zu deaktivieren. Das Deaktivieren eines KMS-Schlüssels verhindert, dass er verwendet wird.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```


Verschlüsselung und Entschlüsselung von AWS KMS Datenschlüsseln mit der Version 3 AWS SDK für PHP

Datenschlüssel sind Verschlüsselungsschlüssel, mit denen Sie Daten verschlüsseln können. Dazu gehören große Datenmengen und andere Datenverschlüsselungsschlüssel.

Sie können AWS Key Management Service an's (AWS KMS) verwenden, [AWS KMS key](#)um Datenschlüssel zu generieren, zu verschlüsseln und zu entschlüsseln.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Verschlüsseln Sie einen Datenschlüssel mit [Encrypt](#).
- Entschlüsseln Sie einen Datenschlüssel mit [Decrypt](#).
- Verschlüsseln Sie einen Datenschlüssel erneut mit einem neuen KMS-Schlüssel mithilfe von [ReEncrypt](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Weitere Informationen zur Verwendung von AWS Key Management Service (AWS KMS) finden Sie im [AWS KMS Entwicklerhandbuch](#).

Encrypt

Die Operation [Encrypt \(Verschlüsseln\)](#) ist für die Verschlüsselung von Datenschlüsseln konzipiert, wird aber nicht häufig verwendet. Die [GenerateDataKeyWithoutPlaintext](#) Operationen [GenerateDataKey](#) und geben verschlüsselte Datenschlüssel zurück. Sie können die Encrypt Methode verwenden, wenn Sie verschlüsselte Daten in eine neue AWS Region verschieben und deren Datenschlüssel mithilfe eines KMS-Schlüssels in der neuen Region verschlüsseln möchten.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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

Zur Entschlüsselung eines Datenschlüssels verwenden Sie die Produktion [Decrypt](#).

Der Wert `CiphertextBlob`, den Sie angeben, muss der Wert des `CiphertextBlob` Felds aus einer [GenerateDataKeyGenerateDataKeyWithoutPlaintext](#), oder [Encrypt-Antwort](#) sein.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Neuverschlüsseln

Verwenden Sie den Vorgang, um einen verschlüsselten Datenschlüssel zu entschlüsseln und den Datenschlüssel dann sofort unter einem anderen KMS-Schlüssel erneut zu verschlüsseln. [ReEncrypt](#) Die Operationen werden ausschließlich serverseitig innerhalb von ausgeführt AWS KMS, sodass Ihr Klartext niemals außerhalb von angezeigt wird. AWS KMS

Der Wert `ciphertextBlob`, den Sie angeben, muss der Wert des `CiphertextBlob` Felds aus einer [GenerateDataKeyGenerateDataKeyWithoutPlaintext](#), oder [Encrypt-Antwort](#) sein.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Mit der AWS SDK für PHP Version 3 mit AWS KMS wichtigen Richtlinien arbeiten

Wenn Sie einen erstellen [AWS KMS key](#), legen Sie fest, wer diesen KMS-Schlüssel verwenden und verwalten kann. Diese Berechtigungen werden in einem Dokument namens Schlüsselrichtlinie festgehalten. Sie können die Schlüsselrichtlinie verwenden, um jederzeit Berechtigungen für einen vom Kunden verwalteten KMS-Schlüssel hinzuzufügen, zu entfernen oder zu ändern, aber Sie können die Schlüsselrichtlinie für einen AWS verwalteten KMS-Schlüssel nicht bearbeiten. Weitere Informationen finden Sie unter [Authentifizierung und Zugriffskontrolle für AWS KMS](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Listet die Namen der wichtigsten Richtlinien auf, die verwendet [ListKeyPolicies](#) werden.
- Holen Sie sich eine wichtige Richtlinie mit [GetKeyPolicy](#).
- Legen Sie eine wichtige Richtlinie fest mit [PutKeyPolicy](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von AWS Key Management Service (AWS KMS) finden Sie im [AWS KMS Entwicklerhandbuch](#).

Alle wichtigen Richtlinien auflisten

Verwenden Sie den `ListKeyPolicies` Vorgang, um die Namen der wichtigsten Richtlinien für einen KMS-Schlüssel abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

```
}
```

Rufen Sie eine wichtige Richtlinie ab

Verwenden Sie den `GetKeyPolicy` Vorgang, um die Schlüsselrichtlinie für einen KMS-Schlüssel abzurufen.

`GetKeyPolicy` erfordert einen Richtliniennamen. Sofern Sie beim Erstellen des KMS-Schlüssels keine Schlüsselrichtlinie erstellt haben, ist der einzig gültige Richtlinienname der Standardwert. Weitere Informationen zur [Standardschlüsselrichtlinie](#) finden Sie im AWS Key Management Service Entwicklerhandbuch.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

```
}
```

Legen Sie eine wichtige Richtlinie fest

Verwenden Sie den `PutKeyPolicy` Vorgang, um eine Schlüsselrichtlinie für einen KMS-Schlüssel einzurichten oder zu ändern.

`PutKeyPolicy` erfordert einen Richtliniennamen. Sofern Sie beim Erstellen des KMS-Schlüssels keine Schlüsselrichtlinie erstellt haben, ist der einzig gültige Richtlinienname der Standardwert. Weitere Informationen zur [Standardschlüsselrichtlinie](#) finden Sie im AWS Key Management Service Entwicklerhandbuch.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}

```

Arbeiten mit Zuschüssen mithilfe der AWS KMS API und der AWS SDK für PHP Version 3

Eine Erteilung ist ein weiterer Mechanismus für die Bereitstellung von Berechtigungen. Es ist eine Alternative zur wichtigsten Richtlinie. Sie können Zuschüsse verwenden, um langfristigen Zugriff zu gewähren, sodass AWS Schulleiter Ihr AWS Key Management Service (AWS KMS) vom Kunden [AWS KMS keys](#) verwaltetes () nutzen können. Weitere Informationen finden Sie unter [Grants AWS KMS im AWS Key Management Service](#) Developer Guide.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Grant für einen KMS-Schlüssel mithilfe von [CreateGrant](#).
- Einen Zuschuss für einen KMS-Schlüssel anzeigen mit [ListGrants](#).
- Einen Zuschuss für einen KMS-Schlüssel zurückziehen mit [RetireGrant](#).

- Widerrufen Sie eine Erteilung für einen KMS-Schlüssel mithilfe von [RevokeGrant](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Weitere Informationen zur Verwendung von AWS Key Management Service (AWS KMS) finden Sie im [AWS KMS Entwicklerhandbuch](#).

Erstellen einer Erteilung

Verwenden Sie die [CreateGrant](#) Operation AWS KMS key, um einen Zuschuss für einen zu erstellen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Anzeigen einer Gewährung

Um detaillierte Informationen zu den Zuschüssen für eine zu erhalten AWS KMS key, verwenden Sie die [ListGrants](#) Operation.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";  
}
```

Einen Zuschuss zurückziehen

Um einen Zuschuss für einen zurückzuziehen AWS KMS key, verwenden Sie den [RetireGrant](#)Vorgang. Heben Sie nicht mehr benötigte Erteilungen auf.

Importe

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

Beispiel-Code

```
$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";
}
```

Widerrufen Sie einen Zuschuss

Um einen Zuschuss für einen zu widerrufen AWS KMS key, verwenden Sie den [RevokeGrant](#)Vorgang. Sie können eine Erteilung widerrufen, um ausdrücklich Produktionen abzulehnen, die diese Erteilung unbedingt benötigen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";  
}
```

Arbeiten mit Aliasen unter Verwendung der AWS KMS API und der AWS SDK für PHP Version 3

AWS Key Management Service (AWS KMS) bietet einen optionalen Anzeigenamen für einen [AWS KMS key](#) aufgerufenen Alias.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Alias mit [CreateAlias](#).
- Einen Alias anzeigen mit [ListAliases](#).
- Aktualisieren Sie einen Alias mit [UpdateAlias](#).
- Löschen Sie einen Alias mit [DeleteAlias](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von AWS Key Management Service (AWS KMS) finden Sie im [AWS KMS Entwicklerhandbuch](#).

Erstellen eines Alias

Verwenden Sie den [CreateAlias](#) Vorgang, um einen Alias für einen KMS-Schlüssel zu erstellen. Der Alias muss für das Konto und die AWS Region eindeutig sein. Wenn Sie einen Alias für einen KMS-Schlüssel erstellen, der bereits über einen Alias verfügt, `CreateAlias` wird ein weiterer Alias für denselben KMS-Schlüssel erstellt. Der vorhandene Alias wird nicht ersetzt.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Einen Alias anzeigen

Verwenden Sie die Operation, um alle Aliase im AWS-Konto und AWS-Region des Anrufers aufzulisten. [ListAliases](#)

Importe

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

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

Beispiel-Code

```
$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";
}
```

Aktualisieren Sie einen Alias

Verwenden Sie den [UpdateAlias](#) Vorgang, um einen vorhandenen Alias einem anderen KMS-Schlüssel zuzuordnen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie einen Alias

Verwenden Sie den [DeleteAlias](#) Vorgang, um einen Alias zu löschen. Das Löschen eines Alias hat keine Auswirkungen auf den zugrunde liegenden KMS-Schlüssel.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Amazon Kinesis Kinesis-Beispiele mit AWS SDK für PHP Version 3

Amazon Kinesis ist ein AWS Service, der Daten in Echtzeit sammelt, verarbeitet und analysiert. Konfigurieren Sie Ihre Datenstreams mit Amazon Kinesis Data Streams oder verwenden Sie Amazon Data Firehose, um Daten an Amazon S3, OpenSearch Service, Amazon Redshift oder Splunk zu senden.

Weitere Informationen zu Kinesis finden Sie in der [Amazon Kinesis Kinesis-Dokumentation](#).

Der gesamte Beispielcode für AWS SDK für PHP Version 3 ist [hier verfügbar](#). [GitHub](#)

Themen

- [Erstellen von Datenströmen mit der Kinesis Data Streams API und der AWS SDK für PHP Version 3](#)
- [Datensplitter mithilfe der Kinesis Data Streams API und der AWS SDK für PHP Version 3 verwalten](#)
- [Erstellen von Lieferdatenströmen mithilfe der Firehose-API und der AWS SDK für PHP Version 3](#)

Erstellen von Datenströmen mit der Kinesis Data Streams API und der AWS SDK für PHP Version 3

Mit Amazon Kinesis Data Streams können Sie Echtzeitdaten senden. Erstellen Sie mit Kinesis Data Streams einen Datenproduzenten, der jedes Mal, wenn Sie Daten hinzufügen, Daten an das konfigurierte Ziel liefert.

Weitere Informationen finden Sie unter [Creating and Managing Streams](#) im Amazon Kinesis Developer Guide.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Datenstream mit [CreateAlias](#).
- Erfahren Sie mehr über einen einzelnen Datenstrom mithilfe von [DescribeStream](#).
- Listet bestehende Datenströme auf mit [ListStreams](#).
- Senden Sie Daten an einen vorhandenen Datenstrom mit [PutRecord](#).
- Löschen Sie einen Datenstrom mit [DeleteStream](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Weitere Informationen zur Verwendung des Amazon Kinesis Developer Guide finden Sie im [Amazon Kinesis Data Streams Developer](#) Guide.

Erstellen Sie einen Datenstream mithilfe eines Kinesis-Datenstroms

Richten Sie mithilfe des folgenden Codebeispiels einen Kinesis-Datenstream ein, an den Sie Informationen senden können, die von Kinesis verarbeitet werden sollen. Weitere Informationen zum [Erstellen und Aktualisieren von Datenströmen](#) finden Sie im Amazon Kinesis Developer Guide.

Verwenden Sie die [CreateStream](#) Operation, um einen Kinesis-Datenstream zu erstellen.

Importe

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

Beispiel-Code

```
$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";
}
```

Rufen Sie einen Datenstrom ab

Rufen Sie mithilfe des folgenden Codebeispiels Details über einen vorhandenen Daten-Stream ab. Standardmäßig werden dadurch Informationen über die ersten 10 Shards zurückgegeben, die mit dem angegebenen Kinesis-Datenstrom verbunden sind. Denken Sie daran, `StreamStatus` anhand der Antwort zu überprüfen, bevor Sie Daten in einen Kinesis-Datenstrom schreiben.

Verwenden Sie den [DescribeStream](#) Vorgang, um Details zu einem bestimmten Kinesis-Datenstrom abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Listet vorhandene Datenströme auf, die mit Kinesis verbunden sind

Listet die ersten 10 Datenströme aus Ihrem Datenstrom AWS-Konto in der ausgewählten AWS Region auf. Verwenden Sie den zurückgegebenen Code `HasMoreStreams`, um zu bestimmen, ob Ihrem Konto weitere Streams zugeordnet sind.

Verwenden Sie den [ListStreams](#) Vorgang, um Ihre Kinesis-Datenströme aufzulisten.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Senden Sie Daten an einen vorhandenen Datenstrom

Sobald Sie einen Daten-Stream erstellt haben, verwenden Sie das folgende Beispiel zum Senden von Daten. Bevor Sie Daten an den Stream senden, überprüfen Sie mithilfe von `DescribeStream`, ob `StreamStatus` für die Daten aktiv ist.

Verwenden Sie die [PutRecord](#) Operation, um einen einzelnen Datensatz in einen Kinesis-Datenstrom zu schreiben. Verwenden Sie den [PutRecords](#) Vorgang, um bis zu 500 Datensätze in einen Kinesis-Datenstrom zu schreiben.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie einen Datenstrom

In diesem Beispiel wird gezeigt, wie Sie einen Daten-Stream löschen. Durch das Löschen eines Datenstroms werden auch alle Daten gelöscht, die Sie an den Daten-Stream gesendet haben. Aktive Kinesis-Datenstreams wechseln in den Status DELETING, bis das Löschen des Streams abgeschlossen ist. Im Status WIRD GELÖSCHT verarbeitet der Stream weiterhin Daten.

Verwenden Sie den [DeleteStream](#) Vorgang, um einen Kinesis-Datenstrom zu löschen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Datensplitter mithilfe der Kinesis Data Streams API und der AWS SDK für PHP Version 3 verwalten

Mit Amazon Kinesis Data Streams können Sie Echtzeitdaten an einen Endpunkt senden. Die Rate des Datenflusses ist von der Anzahl Shards in Ihrem Stream abhängig.

Sie können 1.000 Datensätze pro Sekunde in einen einzelnen Shard schreiben. Jeder Shard verfügt außerdem über ein Upload-Limit von 1 MiB pro Sekunde. Die Nutzung wird berechnet und pro Shard abgerechnet. Mithilfe dieser Beispiele können Sie also die Datenkapazität und Kosten Ihres Streams verwalten.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Shards in einem Stream auflisten mit. [ListShards](#)
- Fügen Sie die Anzahl der Shards in einem Stream hinzu oder reduzieren Sie die Anzahl der Shards mithilfe von. [UpdateShardCount](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon Kinesis Data Streams finden Sie im [Amazon Kinesis Data Streams Developer Guide](#).

Datenstream-Shards auflisten

Listen Sie die Details von bis zu 100 Shards in einem bestimmten Stream auf.

Verwenden Sie die Operation, um die Shards in einem Kinesis-Datenstrom aufzulisten. [ListShards](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Fügen Sie weitere Datenstrom-Shards hinzu

Wenn Sie mehr Daten-Stream-Shards benötigen, können Sie die aktuelle Anzahl Shards erhöhen. Es wird empfohlen, bei einer Erhöhung die Shard-Anzahl zu verdoppeln. Dadurch wird eine Kopie der einzelnen Shards erstellt, die derzeit verfügbar sind, um Ihre Kapazität zu erhöhen. Sie können die Anzahl Ihrer Shards nur zweimal innerhalb von 24 Stunden verdoppeln.

Denken Sie daran, dass die Abrechnung für die Nutzung von Kinesis Data Streams pro Shard berechnet wird. Wenn die Nachfrage sinkt, empfehlen wir Ihnen, die Anzahl Ihrer Shards um die Hälfte zu reduzieren. Wenn Sie Shards entfernen, können Sie die Shard-Menge nur auf die Hälfte Ihrer aktuellen Shard-Anzahl herabsetzen.

Verwenden Sie die Operation, um die Shard-Anzahl eines Kinesis-Datenstroms zu aktualisieren.

[UpdateShardCount](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Erstellen von Lieferdatenströmen mithilfe der Firehose-API und der AWS SDK für PHP Version 3

Mit Amazon Data Firehose können Sie Echtzeitdaten an andere AWS Dienste wie Amazon Kinesis Data Streams, Amazon S3, Amazon OpenSearch Service (OpenSearch Service) und Amazon Redshift oder an Splunk senden. Erstellen Sie ein Datenproduzent mit Bereitstellungstreams, um bei jedem Hinzufügen von Daten Daten an das konfigurierte Ziel zu senden.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Lieferstream mit [CreateDeliveryStream](#)
- Erfahren Sie mehr über einen einzelnen Lieferstream mithilfe von [DescribeDeliveryStream](#).
- Listen Sie Ihre Lieferstreams auf mit [ListDeliveryStreams](#).
- Senden Sie Daten an einen Lieferstream mit [PutRecord](#).
- Löschen Sie einen Lieferstream mit [DeleteDeliveryStream](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von Amazon Data Firehose finden Sie im [Amazon Kinesis Data Firehose](#) Developer Guide.

Erstellen Sie einen Lieferstream mithilfe eines Kinesis-Datenstroms

Verwenden Sie den [CreateDeliveryStream](#) Vorgang, um einen Lieferstream einzurichten, der Daten in einen vorhandenen Kinesis-Datenstrom einfügt.

Auf diese Weise können Entwickler bestehende Kinesis-Dienste auf Firehose migrieren.

Importe

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

Beispiel-Code

```
$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";
}
```

Erstellen Sie einen Lieferstream mithilfe eines Amazon S3 S3-Buckets

Verwenden Sie den [CreateDeliveryStream](#) Vorgang, um einen Lieferstream einzurichten, der Daten in einen vorhandenen Amazon S3 S3-Bucket überträgt.

Stellen Sie die Zielparameter bereit, wie unter [Zielparameter](#) beschrieben. Stellen Sie dann sicher, dass Sie Firehose Zugriff auf Ihren Amazon S3-Bucket [gewähren, wie unter Kinesis Data Firehose Access to an Amazon S3 Destination](#) gewähren beschrieben.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Erstellen Sie mithilfe von Service einen Lieferstream OpenSearch

Verwenden Sie den [CreateDeliveryStream](#) Vorgang, um einen Firehose-Lieferstream einzurichten, der Daten in einen OpenSearch Service-Cluster einfügt.

Stellen Sie die Zielparameter bereit, wie unter [Zielparameter](#) beschrieben. Stellen Sie sicher, dass Sie Firehose Zugriff auf Ihren OpenSearch Service-Cluster [gewähren, wie unter Kinesis Data Firehose Access to an Amazon](#) ES Destination gewähren beschrieben.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Rufen Sie einen Lieferstream ab

Verwenden Sie den [DescribeDeliveryStream](#) Vorgang, um die Details zu einem Firehose Firehose-Lieferstream abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Listet bestehende Lieferstreams auf, die mit Kinesis Data Streams verbunden sind

Verwenden Sie den Vorgang, um alle vorhandenen Firehose-Lieferdatenströme aufzulisten, die Daten an Kinesis Data Streams senden. [ListDeliveryStreams](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Listet vorhandene Lieferströme auf, die Daten an andere Dienste senden AWS

Verwenden Sie den Vorgang, um alle vorhandenen Firehose-Lieferdatenströme aufzulisten, die Daten an Amazon S3, OpenSearch Service oder Amazon Redshift oder an Splunk senden. [ListDeliveryStreams](#)

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Daten an einen vorhandenen Firehose-Lieferstream senden

Um Daten über einen Firehose-Lieferstream an das angegebene Ziel zu senden, verwenden Sie den [PutRecord](#) Vorgang, nachdem Sie einen Firehose-Lieferstream erstellt haben.

Prüfen Sie vor dem Senden von Daten an einen Firehose-Lieferstream `DescribeDeliveryStream`, ob der Zustellungsstream aktiv ist.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code


```
$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";
}
```

Löschen Sie einen Firehose-Lieferstream

Verwenden Sie den [DeleteDeliveryStreams](#) Vorgang, um einen Firehose-Lieferstream zu löschen. Dadurch werden auch alle Daten gelöscht, die Sie an den Bereitstellungsstream gesendet haben.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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 Beispiele mit der AWS SDK für PHP Version 3

AWS Elemental MediaConvert ist ein dateibasierter Video-Transcodierungsdienst mit Funktionen in Rundfunkqualität. Sie können damit Inhalte für die Übertragung und für die video-on-demand (VOD-) Übertragung über das Internet erstellen. Weitere Informationen finden Sie im [AWS Elemental MediaConvert -Benutzerhandbuch](#).

Die PHP-API für AWS Elemental MediaConvert wird über die *AWS.MediaConvertClient*-Klasse verfügbar gemacht. Weitere Informationen finden Sie [Class: AWS.MediaConvert](#) in der API-Referenz.

Erstellen und verwalten Sie Transcodierungsaufträge in AWS Elemental MediaConvert

In diesem Beispiel verwenden Sie AWS SDK für PHP Version 3, um einen Transcodierungsjob aufzurufen AWS Elemental MediaConvert und zu erstellen. Bevor Sie beginnen, müssen Sie das Eingabevideo in den Amazon S3 S3-Bucket hochladen, den Sie für den Eingabespeicher bereitgestellt haben. [Eine Liste der unterstützten Eingabe-Videocodecs und Container finden Sie im Benutzerhandbuch unter Unterstützte Eingabecodecs und Container.](#) [AWS Elemental MediaConvert](#)

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie Transcodierungsaufträge in. AWS Elemental MediaConvert [CreateJob](#).

- Brecht einen Transcodierungsauftrag aus der AWS Elemental MediaConvert Warteschlange ab. [CancelJob](#)
- Rufen Sie den JSON-Code für einen abgeschlossenen Transcodierungsauftrag ab. [GetJob](#)
- Rufen Sie ein JSON-Array für bis zu 20 der zuletzt erstellten Jobs ab. [ListJobs](#)

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Um auf den MediaConvert Client zuzugreifen, erstellen Sie eine IAM-Rolle, die AWS Elemental MediaConvert Zugriff auf Ihre Eingabedateien und die Amazon S3 S3-Buckets gewährt, in denen Ihre Ausgabedateien gespeichert sind. [Einzelheiten finden Sie unter Einrichten von IAM-Berechtigungen im AWS Elemental MediaConvert Benutzerhandbuch](#).

Erstellen Sie einen Client

Konfigurieren Sie den, AWS SDK für PHP indem Sie einen MediaConvert Client mit der Region für Ihren Code erstellen. In diesem Beispiel ist die Region "us-west-2".

Importe

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

use Aws\Exception\AwsException;
use Aws\MediaConvert\MediaConvertClient;
```

Beispiel-Code

```
$mediaConvertClient = new MediaConvertClient([
    'version' => '2017-08-29',
    'region' => 'us-east-2',
    'profile' => 'default'
]);
```

Definition eines einfachen Transcodierungsauftrags

Erstellen Sie die JSON für die Definition der Transcodierungsauftragsparameter.

Diese Parameter sind detailliert. Sie können die [AWS Elemental MediaConvert Konsole](#) verwenden, um die JSON-Jobparameter zu generieren, indem Sie Ihre Job-Einstellungen in der Konsole auswählen und dann am Ende des Job-Abschnitts die Option Job-JSON anzeigen wählen. Dieses Beispiel enthält die JSON für einen einfachen Auftrag.

Beispiel-Code

```
$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"
]
];
```

Erstellen eines Auftrags

Nachdem Sie die Auftragsparameter-JSON erstellt haben, rufen Sie die `createJob`-Methode auf, indem Sie ein `AWS.MediaConvert service object` aufrufen und die Parameter übergeben. Die ID des erstellten Auftrags wird in den Antwortdaten zurückgegeben.

Beispiel-Code

```
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";
}
```

Rufen Sie einen Job ab

Mit der Auftrags-ID, die von `CreateJob` zurückgegeben wurde, erhalten Sie eine detaillierte Beschreibung der zuletzt ausgeführten Aufträge im JSON-Format.

Beispiel-Code

```
$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";
}
```

Abbrechen eines Auftrags

Mit der Auftrags-ID, die vom CreateJob-Aufruf zurückgegeben wurde, können Sie einen Auftrag in der Warteschlange abbrechen. Bereits begonnene Transcodierungsaufträge können nicht mehr abgebrochen werden.

Beispiel-Code

```
$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";
}
```

Listet die letzten Transcodierungsaufträge auf

Erstellen Sie die Parameter-JSON einschließlich der Werte, um festzulegen, ob die Liste in aufsteigender oder absteigender Reihenfolge sortiert wird. Außerdem legen Sie hier den ARN der zu prüfenden Auftragswarteschlange sowie den Status der einzubeziehenden Aufträge fest. Es werden bis zu 20 Aufträge zurückgegeben. Verwenden Sie die Zeichenfolge „nextToken“ aus dem Ergebnis, um die nächsten 20 Aufträge abzurufen.

Beispiel-Code

```
$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";
}
```

Amazon S3 S3-Beispiele mit AWS SDK für PHP Version 3

Amazon Simple Storage Service (Amazon S3) ist ein Webservice, der hoch skalierbaren Cloud-Speicher bereitstellt. Amazon S3 bietet benutzerfreundlichen Objektspeicher mit einer einfachen Webservice-Schnittstelle zum Speichern und Abrufen beliebiger Datenmengen von überall im Internet.

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Themen

- [Amazon S3 S3-Buckets mit AWS SDK für PHP Version 3 erstellen und verwenden](#)

- [Verwaltung von Amazon S3 S3-Bucket-Zugriffsberechtigungen mit der AWS SDK für PHP Version 3](#)
- [Konfiguration von Amazon S3 S3-Buckets mit AWS SDK für PHP Version 3](#)
- [Mehrteilige Amazon S3 S3-Uploads mit AWS SDK für PHP Version 3 verwenden](#)
- [Vorsignierte Amazon S3 S3-URL mit AWS SDK für PHP Version 3](#)
- [Amazon S3 POSTs mit AWS SDK für PHP Version 3 vorsigniert](#)
- [Verwenden eines Amazon S3 S3-Buckets als statischen Webhost mit AWS SDK für PHP Version 3](#)
- [Arbeiten mit Amazon S3 S3-Bucket-Richtlinien mit der AWS SDK für PHP Version 3](#)
- [Verwenden des S3-Zugriffspunkts ARNs der AWS SDK für PHP Version 3](#)
- [Verwenden Sie Amazon S3 Multiregion Access Points mit der AWS SDK für PHP Version 3](#)

Amazon S3 S3-Buckets mit AWS SDK für PHP Version 3 erstellen und verwenden

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Gibt eine Liste von Buckets zurück, die dem authentifizierten Absender der Anfrage gehören, mit. [ListBuckets](#)
- Erstellen Sie einen neuen Bucket mit. [CreateBucket](#)
- Fügen Sie einem Bucket ein Objekt hinzu mit [PutObject](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Importe

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

use Aws\S3\S3Client;
```

Buckets auflisten

Erstellen Sie eine PHP-Datei, mit folgendem Code: Erstellen Sie zunächst einen AWS.S3-Clientdienst, der die AWS Region und die Version angibt. Rufen Sie dann die `listBuckets` Methode auf, die alle Amazon S3 S3-Buckets zurückgibt, die dem Absender der Anfrage gehören, als Array von Bucket-Strukturen.

Beispiel-Code

```
$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";
}
```

Erstellen eines -Buckets

Erstellen Sie eine PHP-Datei, mit folgendem Code: Erstellen Sie zunächst einen AWS.S3-Client-Service, der die AWS Region und die Version angibt. Rufen Sie dann die Methode `createBucket` mit einem Array als Parameter auf. Das einzige erforderliche Feld ist der Schlüssel „Bucket“ mit einem Zeichenfolgenwert für den zu erstellenden Bucket-Namen. Sie können die AWS Region jedoch mit dem Feld 'CreateBucketConfiguration' angeben. Bei Erfolg gibt diese Methode die Position des Buckets zurück.

Beispiel-Code

```
function createBucket($s3Client, $bucketName)
{
    try {
        $result = $s3Client->createBucket([
            'Bucket' => $bucketName,
        ]);
        return 'The bucket\'s location is: ' .
            $result['Location'] . '. ' .
    }
}
```

```

        'The bucket\'s effective URI is: ' .
        $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e->getAwsErrorMessage();
    }
}

function createTheBucket()
{
    $s3Client = new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2006-03-01'
    ]);

    echo createBucket($s3Client, 'amzn-s3-demo-bucket');
}

// Uncomment the following line to run this code in an AWS account.
// createTheBucket();

```

Legen Sie ein Objekt in einen Eimer

Um Dateien zu Ihrem neuen Bucket hinzuzufügen, erstellen Sie eine PHP-Datei mit dem folgenden Code.

Führen Sie in Ihrer Befehlszeile diese Datei aus und übergeben Sie den Namen des Buckets, in den Sie Ihre Datei hochladen möchten, als Zeichenfolge gefolgt vom vollständigen Dateipfad der hochzuladenden Datei.

Beispiel-Code

```

$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";
}
```

Verwaltung von Amazon S3 S3-Bucket-Zugriffsberechtigungen mit der AWS SDK für PHP Version 3

Zugriffskontrolllisten (ACLs) sind eine der ressourcenbasierten Zugriffsrichtlinienoptionen, mit denen Sie den Zugriff auf Ihre Buckets und Objekte verwalten können. Sie können sie verwenden ACLs , um anderen Konten grundlegende read/write Berechtigungen zu gewähren. AWS Weitere Informationen finden Sie unter [Zugriff verwalten mit ACLs](#).

Das folgende Beispiel zeigt eine Anleitung für:

- Rufen Sie die Zugriffskontrollrichtlinie für einen Bucket ab, indem Sie [GetBucketAcl](#).
- Legen Sie die Berechtigungen für einen Bucket mithilfe ACLs von fest [PutBucketAcl](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Rufen Sie eine Richtlinie für die Zugriffskontrollliste ab und legen Sie sie fest

Importe

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

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

Beispiel-Code

```
// 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";
}

```

Konfiguration von Amazon S3 S3-Buckets mit AWS SDK für PHP Version 3

Cross-Origin Resource Sharing (CORS) bestimmt für Client-Webanwendungen, die in einer Domain geladen sind, eine Möglichkeit zur Interaktion mit Ressourcen in einer anderen Domain. Mit der CORS-Unterstützung in Amazon S3 können Sie umfangreiche clientseitige Webanwendungen mit Amazon S3 erstellen und selektiv den ursprungsübergreifenden Zugriff auf Ihre Amazon S3 S3-Ressourcen zulassen.

Weitere Informationen zur Verwendung der CORS-Konfiguration mit einem Amazon S3 S3-Bucket finden Sie unter [Cross-Origin Resource Sharing \(CORS\)](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Rufen Sie die CORS-Konfiguration für einen Bucket ab mit. [GetBucketCors](#)
- Stellen Sie die CORS-Konfiguration für einen Bucket ein mit. [PutBucketCors](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben](#) [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben](#) [the section called “Installation”](#).

Holen Sie sich die CORS-Konfiguration

Erstellen Sie eine PHP-Datei, mit folgendem Code: Zuerst erstellen Sie einen AWS.S3-Client-Service, dann rufen Sie die `getBucketCors`-Methode auf und geben Sie den Bucket an, dessen CORS-Konfiguration Sie abrufen wollen.

Der einzige erforderliche Parameter ist der Name der ausgewählten Buckets. Wenn der Bucket derzeit über eine CORS-Konfiguration verfügt, wird diese Konfiguration von Amazon S3 als [CORSRules Objekt](#) zurückgegeben.

Importe

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

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

Beispiel-Code

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


Stellen Sie die CORS-Konfiguration ein

Erstellen Sie eine PHP-Datei, mit folgendem Code: Zuerst erstellen Sie einen AWS.S3-Client-Service. Rufen Sie dann die `putBucketCors` Methode auf und geben Sie den Bucket an, dessen CORS-Konfiguration festgelegt werden soll, und das `CORSConfiguration` als [CORSRules JSON-Objekt](#).

Importe

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

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

Beispiel-Code

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

Mehrteilige Amazon S3 S3-Uploads mit AWS SDK für PHP Version 3 verwenden

In einer einzelnen PutObject-Operation können Sie Objekte bis zu einer Größe von 5 GB hochladen. Durch Verwendung der Methoden für mehrteilige Uploads (z. B. CreateMultipartUpload, UploadPart, CompleteMultipartUpload, AbortMultipartUpload) können Sie jedoch Objekte von 5 MB bis 5 TB Größe hochladen.

Das folgende Beispiel zeigt eine Anleitung für:

- Laden Sie ein Objekt auf Amazon S3 hoch, mit [ObjectUploader](#).
- Erstellen Sie einen mehrteiligen Upload für ein Amazon S3 S3-Objekt mit [MultipartUploader](#).
- Kopieren Sie Objekte von einem Amazon S3 S3-Standort an einen anderen mithilfe von [ObjectCopier](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Objekt-Uploader

Wenn Sie sich nicht sicher sind, ob PutObject oder für die Aufgabe am besten geeignet MultipartUploader ist, verwenden Sie. ObjectUploader ObjectUploader lädt je nach Nutzlastgröße eine große Datei mit PutObject oder MultipartUploader auf Amazon S3 hoch.

```
require 'vendor/autoload.php';  
  
use Aws\Exception\MultipartUploadException;  
use Aws\S3\MultipartUploader;  
use Aws\S3\ObjectUploader;
```

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

Beispiel-Code

```
// 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);
```

Konfiguration

Der Konstruktor des `ObjectUploader`-Objekts akzeptiert die folgenden Argumente:

\$client

Das `Aws\ClientInterface`-Objekt für die Ausführung der Übertragungen. Dies sollte eine Instance von `Aws\S3\S3Client` sein.

\$bucket

(string, erforderlich) Name des Buckets, in den das Objekt hochgeladen wird.

\$key

(string, erforderlich) Schlüssel, der für das hochzuladende Objekt verwendet werden soll.

\$body

(mixed, erforderlich) Objektdaten zum Hochladen. Kann eine `StreamInterface`, eine PHP-Stream-Ressource oder eine hochzuladende Datenfolge sein.

\$acl

(string) Zugriffskontrollliste (ACL), die auf das das hochzuladende Objekt gesetzt wird. Standardmäßig werden alle Objekte als privat eingestuft.

\$options

Ein assoziatives Array mit Konfigurationsoptionen für den mehrteiligen Upload. Die folgenden Konfigurationsoptionen sind gültig:

add_content_md5

(bool) Auf true setzen, um die MD5 Prüfsumme für den Upload automatisch zu berechnen.

mup_threshold

(int, Standard:int(16777216)) Die Anzahl der Byte für die Dateigröße. Wenn die Dateigröße diese Grenze überschreitet, wird ein mehrteiliger Upload verwendet.

before_complete

(callable) Callback, der vor der `CompleteMultipartUpload`-Operation aufgerufen wird. Der Callback sollte eine Funktionssignatur haben, die der folgenden ähnelt: `function (Aws`

`\Command $command) {...}` In der [CompleteMultipartUpload API-Referenz](#) finden Sie die Parameter, die Sie dem `CommandInterface` Objekt hinzufügen können.

before_initiate

(callable) Callback, der vor der `CreateMultipartUpload`-Operation aufgerufen wird. Der Callback sollte eine Funktionssignatur haben, die der folgenden ähnelt: `function (Aws \Command $command) {...}`. Das SDK ruft diesen Callback auf, wenn die Dateigröße den Wert überschreitet. `mup_threshold` In der [CreateMultipartUpload API-Referenz](#) finden Sie die Parameter, die Sie dem `CommandInterface` Objekt hinzufügen können.

before_upload

(callable) Callback, der vor allen `PutObject UploadPart` OR-Operationen aufgerufen werden soll. Der Callback sollte eine Funktionssignatur haben, die der folgenden ähnelt: `function (Aws \Command $command) {...}` Das SDK ruft diesen Callback auf, wenn die Dateigröße kleiner oder gleich dem Wert ist. `mup_threshold` In der [PutObject API-Referenz](#) finden Sie die Parameter, die Sie auf die `PutObject` Anfrage anwenden können. Parameter, die für eine `UploadPart` Anfrage gelten, finden Sie in der [UploadPart API-Referenz](#). Das SDK ignoriert alle Parameter, die für den durch das `CommandInterface` Objekt repräsentierten Vorgang nicht relevant sind.

concurrency

(`intStandard: int(3)`) Maximale Anzahl der gleichzeitigen `UploadPart`-Operationen, die während des mehrteiligen Upload zulässig sind.

part_size

(`intStandard: int(5242880)`) Teilegröße in Byte, die bei einem mehrteiligen Upload zu verwenden ist. Der Wert muss zwischen 5 MB und einschließlich 5 GB liegen.

state

(`Aws\Multipart\UploadState`) Ein Objekt, das den Status des mehrteiligen Upload darstellt, und das verwendet wird, um einen vorhergehenden Upload fortzusetzen. Wenn diese Option angegeben wird, werden die `$key` Argumente `$bucket` und und die `part_size` Option ignoriert.

params

Ein assoziatives Array, das Konfigurationsoptionen für jeden Unterbefehl bereitstellt. Zum Beispiel:

```
new ObjectUploader($bucket, $key, $body, $acl, ['params' => ['CacheControl'  
=> <some_value>]])
```

MultipartUploader

Mehrteilige Uploads sind darauf ausgelegt, die Upload-Leistung für größere Objekte zu verbessern. Sie ermöglichen Ihnen, Objekte in Teilen unabhängig, in jeder beliebigen Reihenfolge und parallel hochzuladen.

Amazon S3 S3-Kunden wird empfohlen, mehrteilige Uploads für Objekte mit mehr als 100 MB zu verwenden.

MultipartUploader Objekt

Das SDK hat ein spezielles MultipartUploader-Objekt, das den mehrteiligen Upload vereinfacht.

Importe

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

Beispiel-Code

```
$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";
}
```

Der Uploader erstellt einen Generator von Teiledaten, basierend auf der bereitgestellten Quelle und Konfiguration, und versucht, alle Teile hochzuladen. Wenn einige Teile-Uploads fehlschlagen, lädt der Uploader spätere Teile weiter, bis die gesamten Quelldaten gelesen wurden. Danach versucht der Uploader, die fehlgeschlagenen Teile hochzuladen, oder gibt eine Ausnahme zurück, die Informationen zu den Teilen enthält, die nicht hochgeladen wurden.

Anpassen eines mehrteiligen Uploads

Sie können benutzerdefinierte Optionen für die vom Multipart-Uploader ausgeführten Operationen `CreateMultipartUpload`, `UploadPart` und `CompleteMultipartUpload` festlegen. Dazu verwenden Sie Callbacks, die seinem Konstruktor übergeben werden.

Importe

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

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

Beispiel-Code

```
// 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';
},
]);
```

Manuelle Speicherbereinigung zwischen Teil-Uploads

Wenn Sie bei großen Uploads das Speicherlimit ausreizen, ist die unter Umstände auf zyklische Referenzen zurückzuführen, die vom SDK konfiguriert wurden und noch nicht von der [PHP-Speicherbereinigung](#) gelöscht wurden, als das Speicherlimit erreicht wurde. Manuelles Aufrufen des Bereinigungsalgorithmus zwischen Operationen ermöglicht unter Umständen die Löschung der Zyklen, bevor das Limit erreicht wird. Das folgende Beispiel ruft den Bereinigungsalgorithmus mithilfe eines Callbacks vor jedem Teile-Upload auf. Beachten Sie, dass das Aufrufen der Speicherbereinigung zulasten der Leistung geht und die optimale Nutzung von Ihrem Anwendungsfall und Ihrer Umgebung abhängt.

```
$uploader = new MultipartUploader($client, $source, [
    'bucket' => 'your-bucket',
    'key' => 'your-key',
    'before_upload' => function(\Aws\Command $command) {
        gc_collect_cycles();
    }
]);
```

Wiederherstellung nach Fehlern

Wenn während des mehrteiligen Uploads ein Fehler auftritt, wird eine `MultipartUploadException` geworfen. Diese Ausnahme bietet Zugriff auf das `UploadState`-Objekt, das Informationen über den Fortschritt des mehrteiligen Uploads enthält. Der `UploadState` kann verwendet werden, um einen Upload fortzusetzen, der nicht abgeschlossen werden konnte.

Importe

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Beispiel-Code

```
// 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);
}
```

Das Fortsetzen eines Uploads von einem `UploadState` aus versucht nur, Teile hochzuladen, die noch nicht hochgeladen sind. Das Statusobjekt verfolgt fehlende Teile, auch wenn sie nicht aufeinanderfolgend sind. Der Uploader liest oder durchsucht über die mitgelieferte Quelldatei nach den Bytebereichen, die zu den Teilen gehören, die noch hochgeladen werden müssen.

`UploadState`-Objekte sind serialisierbar. Sie können also auch in einem anderen Prozess einen Upload fortzusetzen. Sie können das `UploadState`-Objekt auch dann abrufen, wenn Sie keine Ausnahme verarbeiten. Dazu rufen Sie `$uploader->getState()` auf.

Important

Streams, die einem `MultipartUploader` als Quelle übergeben werden, werden vor dem Hochladen nicht automatisch auf den Anfang zurückgesetzt. Wenn Sie einen Stream anstelle eines Dateipfades in einer Schleife wie im vorherigen Beispiel verwenden, setzen Sie die Variable `$source` innerhalb des `catch`-Blocks zurück.

Importe

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;
```

Beispiel-Code

```
// 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);
```

Abbrechen eines mehrteiligen Uploads

Ein mehrteiliger Upload kann abgebrochen werden, indem die UploadId aus dem Objekt UploadState abgerufen und an abortMultipartUpload übergeben wird.

```
try {
    $result = $uploader->upload();
} catch (MultipartUploadException $e) {
    // State contains the "Bucket", "Key", and "UploadId"
    $params = $e->getState()->getId();
    $result = $s3Client->abortMultipartUpload($params);
}
```

Asynchrone mehrteilige Uploads

Der Aufruf von upload() des MultipartUploader ist eine blockierende Anfrage. Wenn Sie in einem asynchronen Kontext arbeiten, können Sie ein [Versprechen](#) für den mehrteiligen Upload erhalten.

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

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

Beispiel-Code

```
// 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();
```

Konfiguration

Der Konstruktor des `MultipartUploader`-Objekts akzeptiert die folgenden Argumente:

\$client

Das `Aws\ClientInterface`-Objekt für die Ausführung der Übertragungen. Dies sollte eine Instance von `Aws\S3\S3Client` sein.

\$source

Die hochzuladenden Quelldaten. Hierbei kann es sich um einen Pfad oder eine URL (z. B. `/path/to/file.jpg`), einen Ressourcen-Handle (z. B. `fopen('/path/to/file.jpg', 'r')`) oder eine Instance von einem [PSR-7-Stream](#) handeln.

\$config

Ein assoziatives Array mit Konfigurationsoptionen für den mehrteiligen Upload.

Die folgenden Konfigurationsoptionen sind gültig:

acl

(string) Zugriffskontrollliste (ACL), die auf das hochzuladende Objekt gesetzt wird. Standardmäßig werden alle Objekte als privat eingestuft.

before_complete

(callable) Callback, der vor der CompleteMultipartUpload-Operation aufgerufen wird. Der Callback sollte eine Funktionssignatur wie `function (Aws\Command $command) {...}` haben.

before_initiate

(callable) Callback, der vor der CreateMultipartUpload-Operation aufgerufen wird. Der Callback sollte eine Funktionssignatur wie `function (Aws\Command $command) {...}` haben.

before_upload

(callable) Callback, der vor allen UploadPart-Operation aufgerufen wird. Der Callback sollte eine Funktionssignatur wie `function (Aws\Command $command) {...}` haben.

bucket

(string, erforderlich) Name des Buckets, in den das Objekt hochgeladen wird.

concurrency

(intStandard: `int(5)`) Maximale Anzahl der gleichzeitigen UploadPart-Operationen, die während des mehrteiligen Upload zulässig sind.

key

(string, erforderlich) Schlüssel, der für das hochzuladende Objekt verwendet werden soll.

part_size

(intStandard: `int(5242880)`) Teilegröße in Byte, die bei einem mehrteiligen Upload zu verwenden ist. Diese muss zwischen einschließlich 5 MB und 5 GB liegen.

state

(`Aws\Multipart\UploadState`) Ein Objekt, das den Status des mehrteiligen Upload darstellt, und das verwendet wird, um einen vorhergehenden Upload fortzusetzen. Wenn diese Option angegeben ist, werden die Optionen `bucket`, `key` und `part_size` ignoriert.

add_content_md5

(boolean) Auf `true` setzen, um die MD5 Prüfsumme für den Upload automatisch zu berechnen.

params

Ein assoziatives Array, das Konfigurationsoptionen für jeden Unterbefehl bereitstellt. Zum Beispiel:

```
new MultipartUploader($client, $source, ['params' => ['CacheControl'
=> <some_value>]])
```

Mehrteilige Kopien

Das beinhaltet AWS SDK für PHP auch ein `MultipartCopy` Objekt, das ähnlich wie das verwendet wird `MultipartUploader`, aber für das Kopieren von Objekten mit einer Größe zwischen 5 GB und 5 TB innerhalb von Amazon S3 konzipiert ist.

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

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartCopy;
use Aws\S3\S3Client;
```

Beispiel-Code

```
// 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";
}
```

```
}
```

Vorsignierte Amazon S3 S3-URL mit AWS SDK für PHP Version 3

Einige Anfragetypen können Sie authentifizieren, indem Sie die angeforderten Informationen als Abfragezeichenfolgenparameter übergeben, statt den Authentifizierungs-HTTP-Header zu verwenden. Dies ist nützlich, um den direkten Browserzugriff von Drittanbietern auf Ihre privaten Amazon S3 S3-Daten zu ermöglichen, ohne die Anfrage weiterzuleiten. Die Idee ist, eine „vorsignierte“ Anfrage zu erstellen und sie als URL zu codieren, die ein anderer Benutzer verwenden kann. Darüber hinaus können Sie eine vorab signierte Anfrage durch Angabe einer Ablaufzeit begrenzen.

Erstellen Sie eine vorsignierte URL für eine HTTP-GET-Anfrage

Das folgende Codebeispiel zeigt, wie Sie mithilfe des SDK for PHP eine vorsignierte URL für eine HTTP-GET-Anfrage erstellen.

```
<?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;
```

Die [API-Referenz für die `createPresignedRequest` Methode](#) enthält weitere Details.

Eine andere Person kann den `$presignedUrl` Wert verwenden, um das Objekt innerhalb der nächsten Stunde abzurufen. Wenn die HTTP-GET-Anfrage gestellt wird, z. B. über einen Browser, hat der S3-Dienst den Eindruck, dass der Anruf von dem Benutzer kommt, der die vorsignierte URL erstellt hat.

Erstellen Sie eine vorsignierte URL für eine HTTP-PUT-Anfrage

Das folgende Codebeispiel zeigt, wie Sie mithilfe des SDK for PHP eine vorsignierte URL für eine HTTP-PUT-Anfrage erstellen.

```
<?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();
```

Jemand anderes kann jetzt die vorsignierte URL in einer HTTP-PUT-Anfrage verwenden, um eine Datei hochzuladen:

```
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 POSTs mit AWS SDK für PHP Version 3 vorsigniert

Ähnlich wie vorsigniert URLs können Sie mit POSTs vorsignierten Benutzern Schreibzugriff gewähren, ohne ihnen Anmeldeinformationen zu geben. AWS [Vorsignierte POST-Formulare können mit Hilfe einer Instanz von AWSS3 V4 erstellt werden. PostObject](#)

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- [Rufen Sie mit V4 Daten für ein POST-Upload-Formular für S3-Objekte ab. PostObject](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Note

`PostObjectV4` funktioniert nicht mit Anmeldeinformationen, die von stammen AWS IAM Identity Center.

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Erstellen Sie `PostObject V4`

Um eine Instance von `PostObjectV4` zu erstellen, müssen Sie Folgendes bereitstellen:

- Instance von `Aws\S3\S3Client`
- bucket
- assoziatives Array mit Formular-Eingabefeldern
- eine Reihe von Versicherungsbedingungen (siehe [Richtlinienkonstruktion](#) im Amazon Simple Storage Service-Benutzerhandbuch)
- Zeichenfolge für die Ablaufzeit für die Richtlinie (optional, standardmäßig eine Stunde).

Importe

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

use Aws\S3\PostObjectV4;
use Aws\S3\S3Client;
```

Beispiel-Code

```
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'] ?&gt;" method="<?php echo
    $formAttributes['method'] ?&gt;"
    enctype="<?php echo $formAttributes['enctype'] ?&gt;"&gt;
    &lt;label id="key"&gt;</pre
```

```

        <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>

```

Verwenden eines Amazon S3 S3-Buckets als statischen Webhost mit AWS SDK für PHP Version 3

Sie können in Amazon S3 eine statische Website hosten. Weitere Informationen finden Sie unter [Hosten einer statischen Website auf Amazon S3](#).

Das folgende Beispiel zeigt eine Anleitung für:

- Rufen Sie die Website-Konfiguration für einen Bucket mithilfe von [ab GetBucketWebsite](#).
- Stellen Sie die Website-Konfiguration für einen Bucket mithilfe von [ein PutBucketWebsite](#).
- Entfernen Sie die Website-Konfiguration aus einem Bucket mit [DeleteBucketWebsite](#).

Der gesamte Beispielcode für die AWS SDK für PHP Version 3 ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen. Siehe [Anmeldeinformationen für AWS SDK für PHP Version 3](#).

Rufen Sie die Website-Konfiguration für einen Bucket ab, legen Sie sie fest und löschen Sie sie

Importe

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

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

Beispiel-Code

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

// Retrieving the Bucket Website Configuration
$bucket = 'amzn-s3-demo-bucket';
try {
    $resp = $s3Client->getBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in retrieving website configuration for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

```
// Setting a Bucket Website Configuration
$params = [
    'Bucket' => $bucket,
    'WebsiteConfiguration' => [
        'ErrorDocument' => [
            'Key' => 'foo',
        ],
        'IndexDocument' => [
            'Suffix' => 'bar',
        ],
    ],
];

try {
    $resp = $s3Client->putBucketWebsite($params);
    echo "Succeed in setting bucket website configuration.\n";
} catch (AwsException $e) {
    // Display error message
    echo $e->getMessage();
    echo "\n";
}

// Deleting a Bucket Website Configuration
try {
    $resp = $s3Client->deleteBucketWebsite([
        'Bucket' => $bucket
    ]);
    echo "Succeed in deleting policy for bucket: " . $bucket . "\n";
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo "\n";
}
```

Arbeiten mit Amazon S3 S3-Bucket-Richtlinien mit der AWS SDK für PHP Version 3

Sie können eine Bucket-Richtlinie verwenden, um Berechtigungen für Ihre Amazon S3 S3-Ressourcen zu erteilen. Weitere Informationen dazu erhalten Sie unter [Verwendung von Bucket-Richtlinien und Benutzerrichtlinien](#).

Das folgende Beispiel zeigt eine Anleitung für:

- Geben Sie die Richtlinie für einen bestimmten Bucket zurück mit [GetBucketPolicy](#).
- Ersetzen Sie eine Richtlinie für einen Bucket mit [PutBucketPolicy](#).
- Löschen Sie eine Richtlinie aus einem Bucket mithilfe von [DeleteBucketPolicy](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Rufen Sie eine Richtlinie für einen Bucket ab, löschen Sie sie und ersetzen Sie sie

Importe

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

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

Beispiel-Code

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

$bucket = 'amzn-s3-demo-bucket';

// Get the policy of a specific bucket
try {
    $resp = $s3Client->getBucketPolicy([
        'Bucket' => $bucket
    ]);
    echo "Succeed in receiving bucket policy:\n";
    echo $resp->get('Policy');
```

```
        echo "\n";
    } catch (AwsException $e) {
        // Display error message
        echo $e->getMessage();
        echo "\n";
    }

    // Deletes the policy from the bucket
    try {
        $resp = $s3Client->deleteBucketPolicy([
            'Bucket' => $bucket
        ]);
        echo "Succeed in deleting policy of bucket: " . $bucket . "\n";
    } catch (AwsException $e) {
        // Display error message
        echo $e->getMessage();
        echo "\n";
    }

    // Replaces a policy on the bucket
    try {
        $resp = $s3Client->putBucketPolicy([
            'Bucket' => $bucket,
            'Policy' => 'foo policy',
        ]);
        echo "Succeed in put a policy on bucket: " . $bucket . "\n";
    } catch (AwsException $e) {
        // Display error message
        echo $e->getMessage();
        echo "\n";
    }
}
```

Verwenden des S3-Zugriffspunkts ARNs der AWS SDK für PHP Version 3

S3 hat erstmalig Zugriffspunkte verwendet, eine neue Möglichkeit der Interaktion mit S3-Buckets. Auf Zugriffspunkte können eindeutige Richtlinien und Konfigurationen angewendet werden, anstatt dass sie direkt auf den Bucket angewendet werden. Das AWS SDK für PHP ermöglicht es Ihnen, den Access Point ARNs im Bucket-Feld für API-Operationen zu verwenden, anstatt den Bucket-Namen explizit anzugeben. Weitere Informationen darüber, wie S3-Zugangspunkte und ARNs deren Funktionsweise funktionieren, finden Sie [hier](#). In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Wird [GetObject](#) zusammen mit einem Access Point-ARN verwendet, um ein Objekt aus einem Bucket abzurufen.
- Wird [PutObject](#) zusammen mit einem Access Point-ARN verwendet, um ein Objekt zu einem Bucket hinzuzufügen.
- Konfigurieren Sie den S3-Client so, dass er die ARN-Region anstelle der Client-Region verwendet.

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Importe

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

use Aws\S3\S3Client;
```

Objekt abrufen

Erstellen Sie zunächst einen AWS.S3-Clientdienst, der die AWS Region und die Version angibt. Rufen Sie dann die `getObject`-Methode mit Ihrem Schlüssel und einem S3-Zugriffspunkt-ARN im Feld `Bucket` auf. Dadurch wird das Objekt aus dem diesem Zugriffspunkt zugeordneten Bucket abgerufen.

Beispiel-Code

```
$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'
]);
```

Legen Sie ein Objekt in einen Bucket

Erstellen Sie zunächst einen AWS.S3-Client-Dienst, der die AWS Region und die Version angibt. Rufen Sie dann die `putObject`-Methode mit dem gewünschten Schlüssel, dem Hauptteil oder der Quelldatei und einem S3-Zugriffspunkt-ARN im Feld `Bucket` auf. Dadurch wird das Objekt im diesem Zugriffspunkt zugeordneten Bucket abgelegt.

Beispiel-Code

```
$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'
]);
```

Konfigurieren Sie den S3-Client so, dass er die ARN-Region anstelle der Client-Region verwendet.

Wenn Sie einen S3-Zugriffspunkt-ARN in einer S3-Client-Operation verwenden, stellt der Client standardmäßig sicher, dass die ARN-Region mit der Client-Region übereinstimmt. Ist dies nicht der Fall, löst er eine Ausnahme aus. Dieses Verhalten kann geändert werden, um die ARN-Region gegenüber der Client-Region zu bevorzugen, indem Sie für die Konfigurationsoption `use_arn_region` `true` festlegen. Standardmäßig ist für die Option `false` festgelegt.

Beispiel-Code

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

Der Client überprüft auch eine Umgebungsvariable und eine Konfigurationsdateioption in der folgenden Reihenfolge der Priorität:

1. Die Client-Option `use_arn_region`, wie im obigen Beispiel
2. Die Umgebungsvariable `AWS_S3_USE_ARN_REGION`

```
export AWS_S3_USE_ARN_REGION=true
```

1. Die Konfigurationsvariable `s3_use_arn_region` in der AWS gemeinsam genutzten Konfigurationsdatei (standardmäßig in `~/.aws/config`).

```
[default]
s3_use_arn_region = true
```

Verwenden Sie Amazon S3 Multiregion Access Points mit der AWS SDK für PHP Version 3

[Amazon Simple Storage Service \(S3\) Multi-Region-Access Points](#) bieten einen globalen Endpunkt für die Weiterleitung von Amazon S3 S3-Anforderungsdatenverkehr zwischen AWS-Regionen diesen.

Sie können Multi-Region-Access Points [mit dem SDK for PHP](#), einem anderen AWS SDK, der [S3-Konsole](#) oder der [AWS CLI](#) erstellen.

Important

Um Multi-Region Access Points mit dem SDK for PHP verwenden zu können, muss in Ihrer PHP-Umgebung die [AWS Common Runtime \(AWS CRT\) -Erweiterung installiert](#) sein.

Wenn Sie einen Access Point mit mehreren Regionen erstellen, generiert Amazon S3 einen Amazon-Ressourcennamen (ARN), der das folgende Format hat:

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Sie können den generierten ARN anstelle eines Bucket-Namens für [getObject\(\)](#) und [putObject\(\)](#) Methoden verwenden.

```
<?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";
```

Verwaltung von Geheimnissen mit der Secrets Manager API und der AWS SDK für PHP Version 3

AWS Secrets Manager speichert und verwaltet gemeinsam genutzte Geheimnisse wie Passwörter, API-Schlüssel und Datenbankanmeldedaten. Mit dem Secrets Manager-Dienst können Entwickler hartcodierte Anmeldeinformationen im bereitgestellten Code durch einen eingebetteten Aufruf von Secrets Manager ersetzen.

Secrets Manager unterstützt nativ die automatische geplante Rotation von Anmeldeinformationen für Amazon Relational Database Service (Amazon RDS) -Datenbanken und erhöht so die Anwendungssicherheit. Secrets Manager kann auch Secrets für andere Datenbanken und Dienste von Drittanbietern nahtlos rotieren, AWS Lambda um dienstspezifische Details zu implementieren.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie ein Geheimnis mit [CreateSecret](#).
- Rufen Sie ein Geheimnis ab mit [GetSecretValue](#).
- Listet alle von Secrets Manager gespeicherten Geheimnisse mit auf [ListSecrets](#).
- Rufen Sie Details zu einem bestimmten Geheimnis ab mit [DescribeSecret](#).
- Aktualisieren Sie ein bestimmtes Geheimnis mit [PutSecretValue](#).
- Richten Sie eine geheime Rotation ein mit [RotateSecret](#).
- Markieren Sie ein Geheimnis zum Löschen mit [DeleteSecret](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Erstellen Sie ein Geheimnis in Secrets Manager

Verwenden Sie die [CreateSecret](#) Operation, um ein Geheimnis in Secrets Manager zu erstellen.

In diesem Beispiel werden ein Benutzername und ein Passwort als JSON-Zeichenfolge gespeichert.

Importe

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Rufen Sie ein Geheimnis aus Secrets Manager ab

Verwenden Sie den [GetSecretValue](#) Vorgang, um den Wert eines in Secrets Manager gespeicherten Geheimnisses abzurufen.

Im folgenden Beispiel secret handelt es sich um eine Zeichenfolge, die den gespeicherten Wert enthält. Wenn der Wert für username ist <<USERNAME>> und der Wert für password ist <<PASSWORD>>, lautet die Ausgabe von secret:

```
{"username": "<<USERNAME>>", "password": "<<PASSWORD>>"}
```

Wird verwendet `json_decode($secret, true)`, um auf die Array-Werte zuzugreifen.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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;
```

Im Secrets Manager gespeicherte Geheimnisse auflisten

Rufen Sie mithilfe der [ListSecrets](#) Operation eine Liste aller Geheimnisse ab, die von Secrets Manager gespeichert wurden.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```


Rufen Sie Details zu einem Geheimnis ab

Gespeicherte Geheimnisse enthalten Metadaten zu Rotationsregeln, den Zeitpunkt des letzten Zugriffs bzw. der letzten Änderung, vom Benutzer erstellte Tags und den Amazon-Ressourcennamen (ARN). Verwenden Sie den [DescribeSecret](#)Vorgang, um die Details eines bestimmten Geheimnisses abzurufen, das in Secrets Manager gespeichert ist.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Den Secret-Wert aktualisieren

Verwenden Sie den [PutSecretValue](#)Vorgang, um einen neuen verschlüsselten geheimen Wert in Secrets Manager zu speichern.

Auf diese Art wird eine neue Version des Geheimnisses erstellt. Wenn bereits eine Version des Geheimnisses vorhanden ist, fügen Sie den Parameter `VersionStages` mit dem Wert `in AWSCURRENT` hinzu, um sicherzustellen, dass der neue Wert verwendet wird, wenn der Wert abgerufen wird.

Importe

```
require 'vendor/autoload.php';
use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Den Wert auf ein vorhandenes Geheimnis im Secrets Manager drehen

Verwenden Sie eine Lambda-Rotationsfunktion und die [RotateSecret](#) Operation, um den Wert eines vorhandenen Geheimnisses, das in Secrets Manager gespeichert ist, zu rotieren.

Bevor Sie beginnen, erstellen Sie eine Lambda-Funktion, um Ihr Geheimnis zu rotieren. Der [AWS Codebeispielkatalog](#) enthält derzeit mehrere Lambda-Codebeispiele für rotierende Amazon RDS-Datenbankanmeldedaten.

Note

Weitere Informationen zur Rotation von Geheimnissen finden Sie unter [Rotating Your AWS Secrets Manager Secrets](#) im AWS Secrets Manager Benutzerhandbuch.

Nachdem Sie Ihre Lambda-Funktion eingerichtet haben, konfigurieren Sie eine neue geheime Rotation.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Wenn eine Rotation konfiguriert ist, können Sie mithilfe der [RotateSecret](#) Operation eine Rotation implementieren.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie ein Geheimnis aus Secrets Manager

Verwenden Sie den [DeleteSecret](#) Vorgang, um ein bestimmtes Geheimnis aus Secrets Manager zu entfernen. Um zu verhindern, dass ein Geheimnis versehentlich gelöscht wird, wird dem Geheimnis automatisch ein `DeletionDate` Stempel hinzugefügt, der ein Wiederherstellungszeitfenster festlegt,

in dem Sie den Löschvorgang rückgängig machen können. Wird kein Wiederherstellungszeitraum angegeben, so beträgt er standardmäßig 30 Tage.

Importe

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

use Aws\SecretsManager\SecretsManagerClient;
use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Ähnliche Informationen

In den AWS SDK für PHP Beispielen werden die folgenden REST-Operationen aus der AWS Secrets Manager API-Referenz verwendet:

- [CreateSecret](#)
- [GetSecretValue](#)
- [ListSecrets](#)
- [DescribeSecret](#)

- [PutSecretValue](#)
- [RotateSecret](#)
- [DeleteSecret](#)

Weitere Informationen zur Verwendung AWS Secrets Manager finden Sie im [AWS Secrets Manager Benutzerhandbuch](#).

Amazon SES SES-Beispiele mit AWS SDK für PHP Version 3

Amazon Simple Email Service (Amazon SES) ist eine E-Mail-Plattform, die Ihnen eine einfache und kostensparende Möglichkeit bietet, E-Mails mit Ihren eigenen E-Mail-Adressen und Domains zu senden und zu empfangen. Weitere Informationen zu Amazon SES finden Sie im [Amazon SES Developer Guide](#).

AWS bietet zwei Versionen des Amazon SES SES-Service und entsprechend bietet das SDK for PHP zwei Versionen des Clients: [SesClient](#) und [SESV2Client](#). Die Funktionen der Clients überschneiden sich in vielen Fällen, obwohl die Art und Weise, wie die Methoden aufgerufen werden, oder die Ergebnisse unterschiedlich sein können. Die beiden bieten APIs auch exklusive Funktionen, sodass Sie beide Clients verwenden können, um auf alle Funktionen zuzugreifen.

Die Beispiele in diesem Abschnitt verwenden alle das Original, `SesClient`.

Der gesamte Beispielcode für die AWS SDK für PHP Version 3 ist [hier verfügbar GitHub](#).

Themen

- [Überprüfung von E-Mail-Identitäten mithilfe der Amazon SES SES-API und der Version 3 AWS SDK für PHP](#)
- [Erstellen von benutzerdefinierten E-Mail-Vorlagen mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3](#)
- [Verwaltung von E-Mail-Filtern mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3](#)
- [E-Mail-Regeln mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3 erstellen und verwalten](#)
- [Überwachung Ihrer Sendeaktivitäten mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3](#)
- [Autorisieren von Absendern mithilfe der Amazon SES SES-API und der Version 3 AWS SDK für PHP](#)

Überprüfung von E-Mail-Identitäten mithilfe der Amazon SES SES-API und der Version 3 AWS SDK für PHP

Wenn Sie Ihr Amazon Simple Email Service (Amazon SES) -Konto zum ersten Mal verwenden, müssen alle Absender und Empfänger in derselben AWS Region verifiziert sein, an die Sie E-Mails senden. Weitere Informationen zum Senden von E-Mails finden Sie unter [Senden von E-Mails mit Amazon SES](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Überprüfen Sie eine E-Mail-Adresse mit [VerifyEmailIdentity](#).
- Überprüfen Sie eine E-Mail-Domain mit [VerifyDomainIdentity](#).
- Listet alle E-Mail-Adressen auf mit [ListIdentities](#).
- Listet alle E-Mail-Domänen auf, die [ListIdentities](#).
- Entfernen Sie eine E-Mail-Adresse mit [DeletelIdentity](#).
- Entfernen Sie eine E-Mail-Domain mit [DeletelIdentity](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Verifizieren Sie eine E-Mail-Adresse

Amazon SES kann E-Mails nur von verifizierten E-Mail-Adressen oder Domains senden. Durch die Bestätigung einer E-Mail-Adresse weisen Sie nach, dass Sie der Eigentümer dieser Adresse sind und Amazon SES erlauben möchten, E-Mails von dieser Adresse aus zu senden.

Wenn Sie das folgende Codebeispiel ausführen, sendet Amazon SES eine E-Mail an die von Ihnen angegebene Adresse. Wenn Sie (oder der Empfänger der E-Mail) auf den Link in der E-Mail klicken, wird die Adresse verifiziert.

Verwenden Sie den [VerifyEmailIdentity](#) Vorgang, um Ihrem Amazon SES SES-Konto eine E-Mail-Adresse hinzuzufügen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Verifizieren Sie eine E-Mail-Domain

Amazon SES kann E-Mails nur von verifizierten E-Mail-Adressen oder Domains senden. Durch das Verifizieren einer Domäne weisen Sie nach, dass Sie der Eigentümer dieser Domäne sind. Wenn Sie eine Domain verifizieren, erlauben Sie Amazon SES, E-Mails von einer beliebigen Adresse auf dieser Domain zu senden.

Wenn Sie das folgende Codebeispiel ausführen, stellt Ihnen Amazon SES ein Verifizierungstoken zur Verfügung. Sie müssen das Token der DNS-Konfiguration Ihrer Domäne hinzufügen. Weitere Informationen finden Sie unter [Verifying a Domain with Amazon SES](#) im Amazon Simple Email Service Developer Guide.

Verwenden Sie den [VerifyDomainIdentity](#) Vorgang, um Ihrem Amazon SES SES-Konto eine sendende Domain hinzuzufügen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

E-Mail-Adressen auflisten

Verwenden Sie den [ListIdentities](#)Vorgang, um unabhängig vom Bestätigungsstatus eine Liste der in der aktuellen AWS Region eingereichten E-Mail-Adressen abzurufen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

E-Mail-Domänen auflisten

Verwenden Sie den [ListIdentities](#)Vorgang, um unabhängig vom Bestätigungsstatus eine Liste der in der aktuellen AWS Region eingereichten E-Mail-Domänen abzurufen.

Importe

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

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

Beispiel-Code

```
$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";
}
```

Löschen Sie eine E-Mail-Adresse

Verwenden Sie den [DeleteIdentity](#) Vorgang, um eine verifizierte E-Mail-Adresse aus der Identitätsliste zu löschen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie eine E-Mail-Domain

Verwenden Sie den [DeleteIdentity](#) Vorgang, um eine verifizierte E-Mail-Domain aus der Liste der verifizierten Identitäten zu löschen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";  
}
```

Erstellen von benutzerdefinierten E-Mail-Vorlagen mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3

Mit Amazon Simple Email Service (Amazon SES) können Sie mithilfe von Vorlagen E-Mails versenden, die für jeden Empfänger personalisiert sind. Vorlagen enthalten eine Betreffzeile und die Text- und HTML-Teile des E-Mail-Textes. Die Betreff- und Textabschnitte können auch eindeutige Werte enthalten, die für jeden Empfänger personalisiert sind.

Weitere Informationen finden Sie unter [Senden personalisierter E-Mails mit Amazon SES](#) im Amazon Simple Email Service Developer Guide.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie eine E-Mail-Vorlage mit [CreateTemplate](#).
- Listet alle E-Mail-Vorlagen auf, die verwenden [ListTemplates](#).
- Rufen Sie eine E-Mail-Vorlage ab mit [GetTemplate](#).
- Aktualisieren Sie eine E-Mail-Vorlage mit [UpdateTemplate](#).
- Entfernen Sie eine E-Mail-Vorlage mit [DeleteTemplate](#).
- Senden Sie eine E-Mail-Vorlage mit [SendTemplatedEmail](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Erstellen einer E-Mail-Vorlage

Verwenden Sie den [CreateTemplate](#) Vorgang, um eine Vorlage für den Versand personalisierter E-Mail-Nachrichten zu erstellen. Die Vorlage kann von jedem Konto verwendet werden, das zum Senden von Nachrichten in der AWS Region berechtigt ist, zu der die Vorlage hinzugefügt wurde.

Note

Amazon SES validiert Ihren HTML-Code nicht. Stellen Sie daher sicher, dass er gültig HtmlPartist, bevor Sie eine E-Mail senden.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Holen Sie sich eine E-Mail-Vorlage

Verwenden Sie den [GetTemplate](#) Vorgang, um den Inhalt einer vorhandenen E-Mail-Vorlage einschließlich Betreffzeile, HTML-Text und Klartext anzuzeigen. Nur TemplateName ist erforderlich.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";  
}
```

Alle E-Mail-Vorlagen auflisten

Verwenden Sie den [ListTemplates](#) Vorgang, um eine Liste aller E-Mail-Vorlagen abzurufen, die mit Ihrer AWS-Konto in der aktuellen AWS Region verknüpft sind.

Importe

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

Beispiel-Code

```
$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";  
}
```

Aktualisieren einer E-Mail-Vorlage

Verwenden Sie den [UpdateTemplate](#) Vorgang, um den Inhalt einer bestimmten E-Mail-Vorlage zu ändern, einschließlich der Betreffzeile, des HTML-Textkörpers und des Klartextes.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen einer E-Mail-Vorlage

Verwenden Sie den [DeleteTemplate](#) Vorgang, um eine bestimmte E-Mail-Vorlage zu entfernen. Alles was Sie brauchen ist die `TemplateName`.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Senden Sie eine E-Mail mit einer Vorlage

Verwenden Sie den [SendTemplatedEmail](#) Vorgang, um eine Vorlage zum Senden einer E-Mail an Empfänger zu verwenden.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Verwaltung von E-Mail-Filtern mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3

Neben dem Senden von E-Mails können Sie mit Amazon Simple Email Service (Amazon SES) auch E-Mails empfangen. Mithilfe eines IP-Adressenfilters können Sie optional angeben, ob E-Mails, die von einer IP-Adresse oder aus einem IP-Adressbereich stammen, akzeptiert oder abgelehnt werden

sollen. Weitere Informationen finden Sie unter [Verwalten von IP-Adressenfilter für den Amazon SES-E-Mail-Empfang](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen E-Mail-Filter mit [CreateReceiptFilter](#).
- Listet alle E-Mail-Filter auf, die verwenden [ListReceiptFilters](#).
- Entfernen Sie einen E-Mail-Filter mit [DeleteReceiptFilter](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Erstellen Sie einen E-Mail-Filter

Verwenden Sie den [CreateReceiptFilter](#) Vorgang, um E-Mails von einer bestimmten IP-Adresse zuzulassen oder zu blockieren. Geben Sie die IP-Adresse bzw. einen Adressbereich und einen eindeutigen Namen für diesen Filter ein.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Alle E-Mail-Filter auflisten

Verwenden Sie den [ListReceiptFilters](#) Vorgang, um die IP-Adressfilter aufzulisten, die mit Ihrem AWS-Konto in der aktuellen AWS Region verknüpft sind.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie einen E-Mail-Filter

Verwenden Sie den [DeleteReceiptFilter](#) Vorgang, um einen vorhandenen Filter für eine bestimmte IP-Adresse zu entfernen. Geben Sie den eindeutigen Filternamen zur Identifizierung des zu löschenden Empfangsfilters an.

Wenn Sie den Adressbereich, der gefiltert wird, ändern müssen, können Sie einen Empfangsfilter löschen und einen neuen erstellen.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

E-Mail-Regeln mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3 erstellen und verwalten

Neben dem Senden von E-Mails können Sie mit Amazon Simple Email Service (Amazon SES) auch E-Mails empfangen. Mit Empfangsregeln können Sie festlegen, was Amazon SES mit E-Mails macht, die es für die E-Mail-Adressen oder Domains erhält, die Sie besitzen. Eine Regel kann E-Mails an andere AWS Dienste senden, einschließlich, aber nicht beschränkt auf Amazon S3, Amazon SNS oder AWS Lambda.

Weitere Informationen finden Sie unter [Empfangsregelsätze für den Amazon SES SES-E-Mail-Empfang verwalten und Empfangsregeln für den Amazon SES SES-E-Mail-Empfang](#) verwalten.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen Empfangsregelsatz mit [CreateReceiptRuleSet](#).
- Erstellen Sie eine Empfangsregel mit [CreateReceiptRule](#).
- Beschreiben Sie einen Empfangsregelsatz mit [DescribeReceiptRuleSet](#).
- Beschreiben Sie eine Empfangsregel mit [DescribeReceiptRule](#).
- Listet alle Empfangsregelsätze auf, die verwenden [ListReceiptRuleSets](#).
- Aktualisieren Sie eine Empfangsregel mit [UpdateReceiptRule](#).
- Entfernen Sie eine Empfangsregel mit [DeleteReceiptRule](#).
- Entfernen Sie einen Empfangsregelsatz mit [DeleteReceiptRuleSet](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Erstellen eines Empfangsregelsatzes

Ein Empfangsregelsatz enthält eine Sammlung von Empfangsregeln. Sie müssen mindestens einen Empfangsregelsatz mit Ihrem Konto verknüpft haben, bevor Sie eine Empfangsregel erstellen können. Um einen Satz von Empfangsregeln zu erstellen, geben Sie einen eindeutigen Wert an `RuleSetName` und verwenden Sie den [CreateReceiptRuleSet](#) Vorgang.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```


Erstellen einer Empfangsregel

Kontrollieren Sie Ihre eingehenden E-Mails durch Hinzufügen einer Empfangsregel zu einem vorhandenen Empfangsregelsatz. Dieses Beispiel zeigt Ihnen, wie Sie eine Empfangsregel erstellen, die eingehende Nachrichten an einen Amazon S3 S3-Bucket sendet. Sie können aber auch Nachrichten an Amazon SNS und AWS Lambda senden. Um eine Empfangsregel zu erstellen, geben Sie eine Regel und dann den [CreateReceiptRule](#)Vorgang RuleSetName an.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Beschreiben eines Empfangsregelsatzes

Die Details des angegebenen Empfangsregelsatzes werden einmal pro Sekunde zurückgegeben. Um die [DescribeReceiptRuleSet](#) Operation zu verwenden, geben Sie die an RuleSetName.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";  
}
```

Beschreiben Sie eine Empfangsregel

Die Details einer angegebenen Empfangsregel werden zurückgegeben. Um die [DescribeReceiptRule](#) Operation zu verwenden, geben Sie RuleName und ein RuleSetName.

Importe

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

Beispiel-Code

```
$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";  
}
```

Listet alle Empfangsregelsätze auf

Verwenden Sie den [ListReceiptRuleSets](#) Vorgang, um die Empfangsregelsätze aufzulisten, die AWS-Konto in Ihrer aktuellen AWS Region existieren.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Aktualisieren Sie eine Empfangsregel

Dieses Beispiel zeigt Ihnen, wie Sie eine Empfangsregel aktualisieren, die eingehende Nachrichten an eine AWS Lambda Funktion sendet. Sie können aber auch Nachrichten an Amazon SNS und Amazon S3 senden. Um den [UpdateReceiptRule](#) Vorgang zu verwenden, geben Sie die neue Empfangsregel und die an RuleSetName.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen Sie einen Empfangsregelsatz

Entfernen Sie einen bestimmten Empfangsregelsatz, der derzeit nicht deaktiviert ist. Mit diesem Vorgang werden auch alle darin enthaltenen Empfangsregeln gelöscht. Um einen Empfangsregelsatz zu löschen, stellen Sie den für RuleSetName den [DeleteReceiptRuleSet](#)Vorgang bereit.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Löschen einer Empfangsregel

Um eine angegebene Empfangsregel zu löschen, geben Sie den RuleName Wert und RuleSetName für den [DeleteReceiptRule](#)Vorgang ein.

Importe

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

use Aws\Exception\AwsException;
```

Beispiel-Code

```
$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";
}
```

Überwachung Ihrer Sendeaktivitäten mithilfe der Amazon SES SES-API und der AWS SDK für PHP Version 3

Amazon Simple Email Service (Amazon SES) bietet Methoden zur Überwachung Ihrer Versandaktivitäten. Am besten implementieren Sie diese Methoden, damit Sie wichtige Maßnahmen – wie Ihre kontobezogenen Quoten für Unzustellbarkeit, Beschwerden und Ablehnungen – verfolgen können. Zu hohe Absprungs- und Beschwerderaten können Ihre Fähigkeit, E-Mails mit Amazon SES zu versenden, gefährden.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Überprüfen Sie Ihr Versandkontingent mit [GetSendQuota](#)
- Überwachen Sie Ihre Sendeaktivität mit [GetSendStatistics](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Überprüfen Sie Ihr Versandkontingent

Sie können nur eine bestimmte Menge an Nachrichten in einem einzelnen 24-Stunden-Zeitraum senden. Verwenden Sie den [GetSendQuota](#) Vorgang, um zu überprüfen, wie viele Nachrichten Sie noch senden dürfen. Weitere Informationen finden Sie unter [Verwalten Ihrer Sendelimits für Amazon SES](#).

Importe

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";
}
```

Überwachen Sie Ihre Sendeaktivität

Verwenden Sie den [GetSendStatistics](#) Vorgang, um Messwerte für Nachrichten abzurufen, die Sie in den letzten zwei Wochen gesendet haben. Dieses Beispiel gibt die Anzahl der Zustellungsversuche, Unzustellbarkeiten, Beschwerden und abgelehnten Nachrichten in 15-Minuten-Schritten zurück.

Importe

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";  
}
```

Autorisieren von Absendern mithilfe der Amazon SES SES-API und der Version 3 AWS SDK für PHP

Um es einem anderen AWS-Konto AWS Identity and Access Management Benutzer oder AWS Dienst zu ermöglichen, in Ihrem Namen E-Mails über Amazon Simple Email Service (Amazon SES) zu senden, erstellen Sie eine Versandautorisierungsrichtlinie. Hierbei handelt es sich um ein JSON-Dokument, das Sie einer in Ihrem Besitz befindlichen Identität anfügen.

Die Richtlinie führt explizit auf, wem Sie die Berechtigung erteilen, für diese Identität E-Mails zu senden und unter welchen Bedingungen. Mit Ausnahme von Ihnen und den Entitäten, denen Sie in der Richtlinie ausdrücklich die Berechtigung erteilen, wird allen anderen Sendern das Senden von E-Mails verweigert. Einer Identität kann keine Richtlinie, eine Richtlinie oder mehrere Richtlinien angefügt werden. Eine Richtlinie kann auch mehrere Anweisungen enthalten und so die gleiche Wirkung wie mehrere Richtlinien erzielen.

Weitere Informationen finden Sie unter [Verwenden der Sendeautorisierung mit Amazon SES](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie einen autorisierten Absender mit [PutIdentityPolicy](#).
- Rufen Sie Richtlinien für einen autorisierten Absender ab mit [GetIdentityPolicies](#).
- Liste autorisierter Absender mit. [ListIdentityPolicies](#)
- Widerrufen Sie die Erlaubnis für einen autorisierten Absender mit [DeleteIdentityPolicy](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Weitere Informationen zur Verwendung von Amazon SES finden Sie im [Amazon SES Developer Guide](#).

Erstellen Sie einen autorisierten Absender

Um eine andere Person AWS-Konto zu autorisieren, in Ihrem Namen E-Mails zu versenden, verwenden Sie eine Identitätsrichtlinie, um die Autorisierung für das Senden von E-Mails von Ihren verifizierten E-Mail-Adressen oder Domains hinzuzufügen oder zu aktualisieren. Verwenden Sie den [PutIdentityPolicy](#) Vorgang, um eine Identitätsrichtlinie zu erstellen.

Importe

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";
}
```

Rufen Sie Richtlinien für einen autorisierten Absender ab

Geben Sie die Sendeautorisierungsrichtlinien, die mit einer bestimmten E-Mail-Identität oder Domänenidentität verknüpft sind, zurück. Verwenden Sie den [GetIdentityPolicy](#) Vorgang, um die Sendeautorisierung für eine bestimmte E-Mail-Adresse oder Domain zu erhalten.

Importe

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";
}
```

Autorisierte Absender auflisten

Verwenden Sie den [ListIdentityPolicies](#) Vorgang, um die Richtlinien für die Sendeautorisierung aufzulisten, die mit einer bestimmten E-Mail-Identität oder Domänenidentität in der aktuellen AWS Region verknüpft sind.

Importe

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

use Aws\Exception\AwsException;
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";  
}
```

Widerrufen Sie die Erlaubnis für einen autorisierten Absender

Entfernen Sie die Sendeautorisierung für einen anderen AWS-Konto Benutzer, E-Mails mit einer E-Mail-Identität oder Domänenidentität zu senden, indem Sie die mit dem [DeleteIdentityPolicy](#) Vorgang verknüpfte Identitätsrichtlinie löschen.

Importe

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Ses\SesClient;
```

Beispiel-Code

```
$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";  
}
```

Amazon SNS SNS-Beispiele mit AWS SDK für PHP Version 3

Amazon Simple Notification Service (Amazon SNS) ist ein Webservice, der die Zustellung oder das Senden von Nachrichten an abonnierende Endpunkte oder Clients koordiniert und verwaltet.

In Amazon SNS gibt es zwei Arten von Kunden: Verlage (auch als Produzenten bezeichnet) und Abonnenten (auch als Verbraucher bezeichnet). Herausgeber kommunizieren asynchron mit Abonnenten, indem sie eine Nachricht erstellen und an ein Thema senden, bei dem es sich wirklich um einen logischen Zugriffspunkt und Kommunikationskanal handelt. Abonnenten (Webserver, E-Mail-Adressen, Amazon SQS-Warteschlangen, AWS Lambda Funktionen) konsumieren oder empfangen die Nachricht oder Benachrichtigung über eines der unterstützten Protokolle (Amazon SQS, HTTP/HTTPS, E-Mail, Lambda) URLs, wenn sie das Thema abonniert haben. AWS SMS

[Der gesamte Beispielcode für die Version 3 ist hier verfügbar. AWS SDK für PHP GitHub](#)

Themen

- [Themen in Amazon SNS mit AWS SDK für PHP Version 3 verwalten](#)
- [Verwaltung von Abonnements in Amazon SNS mit AWS SDK für PHP Version 3](#)
- [Senden von SMS-Nachrichten in Amazon SNS mit der AWS SDK für PHP Version 3](#)

Themen in Amazon SNS mit AWS SDK für PHP Version 3 verwalten

Um Benachrichtigungen an Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS URLs, E-Mail, AWS SMS oder zu senden AWS Lambda, müssen Sie zunächst ein Thema erstellen, das die Zustellung von Nachrichten an alle Abonnenten dieses Themas verwaltet.

Im Hinblick auf das Entwurfsmuster ist ein Thema für den Beobachter mit dem Betreff vergleichbar. Nach dem Erstellen eines Themas können Sie Abonnenten hinzufügen, die automatisch benachrichtigt werden, wenn eine Nachricht im Thema veröffentlicht wird.

Weitere Informationen zum Abonnieren von Themen finden Sie unter [Verwaltung von Abonnements in Amazon SNS mit AWS SDK für PHP Version 3](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Erstellen Sie ein Thema, um Benachrichtigungen zu veröffentlichen, um es zu verwenden. [CreateTopic](#)
- Gibt eine Liste der Themen des Anfragenden zurück, indem Sie [ListTopics](#).
- Löschen Sie ein Thema und alle zugehörigen Abonnements mit [DeleteTopic](#).
- Gibt alle Eigenschaften eines Themas zurück mit [GetTopicAttributes](#).
- Erlauben Sie einem Themenbesitzer, ein Attribut des Themas auf einen neuen Wert zu setzen, indem Sie [SetTopicAttributes](#).

Weitere Informationen zur Verwendung von Amazon SNS finden Sie unter Amazon SNS [SNS-Thema Attribute für den Status der Nachrichtenzustellung](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Erstellen eines Themas

Verwenden Sie die [CreateTopic](#) Operation, um ein Thema zu erstellen.

Jeder Themenname in Ihrem AWS-Konto muss eindeutig sein.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Listen Sie Ihre Themen auf

Verwenden Sie die [ListTopics](#) Operation, um bis zu 100 bestehende Themen in der aktuellen AWS Region aufzulisten.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Löschen eines Themas

Verwenden Sie den [DeleteTopic](#) Vorgang, um ein vorhandenes Thema und alle zugehörigen Abonnements zu entfernen.

Alle Nachrichten, die den Abonnenten noch nicht zugestellt wurden, werden ebenfalls gelöscht.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Themenattribute abrufen

Verwenden Sie den [GetTopicAttributes](#) Vorgang, um Eigenschaften eines einzelnen vorhandenen Themas abzurufen.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Festlegen von Themenattributen

Verwenden Sie den [SetTopicAttributes](#) Vorgang, um die Eigenschaften eines einzelnen vorhandenen Themas zu aktualisieren.

Sie können nur die Attribute Policy, DisplayName und DeliveryPolicy festlegen.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Verwaltung von Abonnements in Amazon SNS mit AWS SDK für PHP Version 3

Verwenden Sie Amazon Simple Notification Service (Amazon SNS) -Themen, um Benachrichtigungen an Amazon Simple Queue Service (Amazon SQS), HTTP/HTTPS, E-Mail-Adressen, AWS Server Migration Service (AWS SMS) oder zu senden. AWS Lambda

Abonnements werden einem Thema angefügt, das das Senden von Nachrichten an Abonnenten verwaltet. Weitere Informationen zum Erstellen von Themen finden Sie [unter Themen in Amazon SNS mit AWS SDK für PHP Version 3 verwalten](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Abonnieren Sie ein vorhandenes Thema mit der Operation [Subscribe](#).
- Überprüfen Sie ein Abonnement mit [ConfirmSubscription](#).
- Bestehende Abonnements auflisten mit [ListSubscriptionsByTopic](#).
- Löschen Sie ein Abonnement mit der Operation [Unsubscribe](#).
- Senden Sie eine Nachricht an alle Abonnenten eines Themas mit der Operation [Publish](#).

Weitere Informationen zur Verwendung von Amazon SNS finden Sie unter [Amazon SNS for System-to-System Messaging verwenden](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben](#) [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben](#) [the section called “Installation”](#).

Abonnieren einer E-Mail-Adresse für ein Thema

Um ein Abonnement für eine E-Mail-Adresse abzuschließen, verwenden Sie die Operation [Subscribe](#).

Sie können die Abonnement-Methode verwenden, um mehrere verschiedene Endpunkte für ein Amazon SNS SNS-Thema zu abonnieren, abhängig von den Werten, die für die übergebenen Parameter verwendet werden. Dies wird in anderen Beispielen in diesem Thema veranschaulicht.

In diesem Beispiel handelt es sich bei dem Endpunkt um eine E-Mail-Adresse. Ein Bestätigungstoken wird an diese E-Mail-Adresse gesendet. Verifizieren Sie das Abonnement mit diesem Bestätigungstoken innerhalb von drei Tagen nach Erhalt.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Abonnieren Sie ein Thema über einen Anwendungsendpunkt

Um ein Abonnement für eine Web-App abzuschließen, verwenden Sie die Operation [Subscribe](#).

Sie können die Abonnement-Methode verwenden, um mehrere verschiedene Endpunkte für ein Amazon SNS SNS-Thema zu abonnieren, abhängig von den Werten, die für die übergebenen Parameter verwendet werden. Dies wird in anderen Beispielen in diesem Thema veranschaulicht.

In diesem Beispiel ist der Endpunkt eine URL. Ein Bestätigungs-Token wird an diese Webadresse gesendet. Verifizieren Sie das Abonnement mit diesem Bestätigungs-Token innerhalb von drei Tagen nach Erhalt.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Abonnieren eines Themas mit einer Lambda-Funktion

Verwenden Sie den Vorgang [Abonnieren](#), um ein Abonnement für eine Lambda-Funktion zu initiieren.

Sie können die Abonnement-Methode verwenden, um mehrere verschiedene Endpunkte für ein Amazon SNS SNS-Thema zu abonnieren, abhängig von den Werten, die für die übergebenen Parameter verwendet werden. Dies wird in anderen Beispielen in diesem Thema veranschaulicht.

In diesem Beispiel ist der Endpunkt eine Lambda-Funktion. Ein Bestätigungstoken wird an diese Lambda-Funktion gesendet. Verifizieren Sie das Abonnement mit diesem Bestätigungs-Token innerhalb von drei Tagen nach Erhalt.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Abonnieren Sie eine Text-SMS zu einem Thema

Um SMS-Nachrichten an mehrere Telefonnummern gleichzeitig zu senden, abonnieren Sie ein Thema mit jeder einzelnen Telefonnummer.

Um ein Abonnement für eine Telefonnummer abzuschließen, verwenden Sie die Operation [Subscribe](#).

Sie können die Abonnement-Methode verwenden, um mehrere verschiedene Endpunkte für ein Amazon SNS SNS-Thema zu abonnieren, abhängig von den Werten, die für die übergebenen Parameter verwendet werden. Dies wird in anderen Beispielen in diesem Thema veranschaulicht.

In diesem Beispiel ist der Endpunkt eine Telefonnummer im E.164-Format, einem Standard für die internationale Schreibweise für Telefonnummern.

Ein Bestätigungs-Token wird an diese Telefonnummer gesendet. Verifizieren Sie das Abonnement mit diesem Bestätigungs-Token innerhalb von drei Tagen nach Erhalt.

Eine alternative Methode zum Senden von SMS-Nachrichten mit Amazon SNS finden Sie unter [Senden von SMS-Nachrichten in Amazon SNS mit AWS SDK für PHP Version 3](#).

Importe


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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Bestätigen Sie das Abonnement für ein Thema

Um ein Abonnement letztendlich zu erstellen, muss der Eigentümer des Endpunkts die Absicht, Nachrichten von dem Thema zu empfangen, mithilfe eines Tokens bestätigen, das beim anfänglichen Abschluss des Abonnements gesendet wurde, wie zuvor beschrieben. Bestätigungs-Token sind drei Tage gültig. Nach drei Tagen können Sie ein Token erneut senden, indem Sie ein neues Abonnement abschließen.

Verwenden Sie den [ConfirmSubscription](#) Vorgang, um ein Abonnement zu bestätigen.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Abonnements für ein Thema auflisten

Verwenden Sie den [ListSubscriptions](#) Vorgang, um bis zu 100 bestehende Abonnements in einer bestimmten AWS Region aufzulisten.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Abbestellen eines Themas

Verwenden Sie zum Entfernen eines Endpunkts, für den ein Thema abonniert wurde, die Operation [Unsubscribe](#).

Wenn für das Löschen des Abonnements eine Authentifizierung erforderlich ist, kann nur der Eigentümer des Abonnements oder der Eigentümer des Themas das Abonnement kündigen. Eine AWS Unterschrift ist erforderlich. Wenn der Kündigungsaufwurf keine Authentifizierung erfordert und der Anforderer nicht der Eigentümer des Abonnements ist, wird eine Nachricht über die endgültige Kündigung des Abonnements an den Endpunkt gesendet.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Eine Nachricht zu einem Amazon SNS SNS-Thema veröffentlichen

Verwenden Sie den Vorgang [Veröffentlichen](#), um eine Nachricht an jeden Endpunkt zu senden, der ein Amazon SNS SNS-Thema abonniert hat.

Erstellen Sie ein Objekt, das die Parameter für die Veröffentlichung einer Nachricht enthält, einschließlich des Nachrichtentextes und des Amazon-Ressourcennamens (ARN) des Amazon SNS-Themas.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Senden von SMS-Nachrichten in Amazon SNS mit der AWS SDK für PHP Version 3

Sie können Amazon Simple Notification Service (Amazon SNS) verwenden, um Textnachrichten oder SMS-Nachrichten an SMS-fähige Geräte zu senden. Sie können eine Nachricht direkt an eine Telefonnummer senden oder Sie können eine Nachricht an mehrere Telefonnummern gleichzeitig senden, indem Sie das Thema für diese Telefonnummern abonnieren und die Nachricht an das Thema senden.

Verwenden Sie Amazon SNS, um Einstellungen für SMS-Nachrichten festzulegen, z. B. wie Ihre Lieferungen optimiert werden (aus Kostengründen oder für eine zuverlässige Zustellung), Ihr monatliches Ausgabenlimit, wie Nachrichtenzustellungen protokolliert werden und ob Sie tägliche SMS-Nutzungsberichte abonnieren möchten. Diese Einstellungen werden abgerufen und als SMS-Attribute für Amazon SNS festgelegt.

Wenn Sie eine SMS-Nachricht senden, geben Sie die Telefonnummer im E.164-Format an. Die Richtlinie E.164 legt die internationale Schreibweise für Telefonnummern fest. Telefonnummern in diesem Format bestehen aus maximal 15 Zeichen sowie einem vorangestellten Plus-Zeichen (+) und der Ländervorwahl. Eine US-Telefonnummer im E.164-Format würde beispielsweise als XXX555 +1001 0100 angezeigt.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- [Rufen Sie mit Get die Standardeinstellungen für das Senden von SMS-Nachrichten von Ihrem Konto aus ab. SMSAttributes](#)
- Aktualisieren Sie mithilfe von [Set](#) die Standardeinstellungen für das Senden von SMS-Nachrichten von Ihrem Konto aus SMSAttributes.

- [Finden Sie mithilfe CheckIfPhoneNumber ISOpted von Out heraus, ob sich der Inhaber einer bestimmten Telefonnummer gegen den Empfang von SMS-Nachrichten von Ihrem Konto entschieden hat.](#)
- Listen Sie Telefonnummern auf, bei denen der Eigentümer den Empfang von SMS-Nachrichten von Ihrem Konto mithilfe von deaktiviert hat [ListPhoneNumberOptedOut](#).
- Senden Sie eine Textnachricht (SMS-Nachricht) mit [Publish](#) direkt an eine Telefonnummer.

Weitere Informationen zur Verwendung von Amazon SNS finden Sie unter [Verwenden von Amazon SNS für Benutzerbenachrichtigungen mit einer Mobiltelefonnummer als Abonnent \(SMS senden\)](#).

Der gesamte Beispielcode für die AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Holen Sie sich SMS-Attribute

Um die Standardeinstellungen für SMS-Nachrichten abzurufen, verwenden Sie den SMSAttributes Vorgang [Abrufen](#).

In diesem Beispiel wird das DefaultSMSType-Attribut abgerufen. Dieses Attribut steuert, ob SMS-Nachrichten als Promotional oder als Transactional gesendet werden. Im ersten Fall wird die Nachrichtenzustellung im Hinblick auf die Kosten und im zweiten Fall im Hinblick auf höchste Zuverlässigkeit optimiert.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

```
$SnsClient = new SnsClient([
    'profile' => 'default',
```

```
'region' => 'us-east-1',
'version' => '2010-03-31'
]);

try {
    $result = $SnSClient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

SMS-Attribute festlegen

Verwenden Sie den `SMSAttributes` Vorgang [Set](#), um die Standardeinstellungen für SMS-Nachrichten zu aktualisieren.

In diesem Beispiel wird das `DefaultSMSType`-Attribut auf `Transactional` festgelegt. Damit wird die Nachrichtenzustellung im Hinblick auf höchste Zuverlässigkeit optimiert.

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Prüfen Sie, ob sich eine Telefonnummer abgemeldet hat

Verwenden Sie den [CheckIfPhoneNumberIsOptedOut](#)Vorgang, um festzustellen, ob der Inhaber einer bestimmten Telefonnummer den Empfang von SMS-Nachrichten von Ihrem Konto abgelehnt hat.

In diesem Beispiel folgt die Telefonnummer dem E.164-Format, einem Standard für die internationale Schreibweise für Telefonnummern.

Importe

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Rufnummern auflisten, von denen Sie sich abgemeldet haben

Verwenden Sie den Vorgang, um eine Liste mit Telefonnummern abzurufen, für die der Eigentümer den Empfang von SMS-Nachrichten von Ihrem Konto deaktiviert hat. [ListPhoneNumbersOptedOut](#)

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

In einer Textnachricht (SMS-Nachricht) veröffentlichen

Um eine Textnachricht (SMS-Nachricht) direkt an eine Telefonnummer zu senden, verwenden Sie die Operation [Publish](#).

In diesem Beispiel folgt die Telefonnummer dem E.164-Format, einem Standard für die internationale Schreibweise für Telefonnummern.

SMS-Nachrichten können bis zu 140 Byte enthalten. Für die veröffentlichte und in mehreren Teilen versendete SMS-Nachricht gilt eine Größenbegrenzung von 1 600 Byte.

Weitere Informationen zum Senden von SMS-Nachrichten finden Sie unter [Senden einer SMS-Nachricht](#).

Importe

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

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

Beispiel-Code

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

Amazon SQS SQS-Beispiele mit AWS SDK für PHP Version 3

Amazon Simple Queue Service (SQS) ist ein schneller, zuverlässiger, skalierbarer und vollständig verwalteter Message Queuing-Service. Mit Amazon SQS können Sie die Komponenten einer Cloud-

Anwendung entkoppeln. Amazon SQS umfasst Standardwarteschlangen mit hohem Durchsatz und hoher at-least-once Verarbeitung sowie FIFO-Warteschlangen, die FIFO-Zustellung (First>In, First>Out) und Exactly-Once-Verarbeitung ermöglichen.

[Der gesamte Beispielcode für Version 3 ist hier verfügbar. AWS SDK für PHP GitHub](#)

Themen

- [Aktivierung von Long Polling in Amazon SQS mit Version 3 AWS SDK für PHP](#)
- [Verwaltung des Sichtbarkeits-Timeouts in Amazon SQS mit Version 3 AWS SDK für PHP](#)
- [Senden und Empfangen von Nachrichten in Amazon SQS mit AWS SDK für PHP Version 3](#)
- [Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS mit Version 3 AWS SDK für PHP](#)
- [Verwenden von Warteschlangen in Amazon SQS mit Version 3 AWS SDK für PHP](#)

Aktivierung von Long Polling in Amazon SQS mit Version 3 AWS SDK für PHP

Lange Abfragen reduzieren die Anzahl leerer Antworten, indem Amazon SQS eine bestimmte Zeit warten kann, bis eine Nachricht in der Warteschlange verfügbar ist, bevor eine Antwort gesendet wird. Durch Langabfragen lassen sich außerdem falsch leere Antworten vermeiden, indem die Anfrage statt an eine Auswahl an Servern an alle Server gesendet wird. Zum Aktivieren von Langabfragen geben Sie für empfangene Nachrichten eine Wartezeit ungleich Null an. Weitere Informationen finden Sie unter [SQS Langabfragen](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Legen Sie Attribute in einer Amazon SQS SQS-Warteschlange fest, um lange Abfragen zu ermöglichen, mithilfe von. [SetQueueAttributes](#)
- Rufen Sie eine oder mehrere Nachrichten mit langer Abfrage ab mit. [ReceiveMessage](#)
- Erstellen Sie eine lange Abfragewarteschlange mit. [CreateQueue](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Legen Sie Attribute für eine Warteschlange fest, um lange Abfragen zu ermöglichen

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Rufen Sie Nachrichten mit langen Abfragen ab

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Erstellen Sie eine Warteschlange mit langen Abfragen

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Verwaltung des Sichtbarkeits-Timeouts in Amazon SQS mit Version 3 AWS SDK für PHP

Ein Sichtbarkeits-Timeout ist ein Zeitraum, in dem Amazon SQS verhindert, dass andere verbrauchende Komponenten eine Nachricht empfangen und verarbeiten. Weitere Informationen finden Sie unter [Zeitbeschränkung für die Sichtbarkeit](#).

Das folgende Beispiel zeigt eine Anleitung für:

- Ändern Sie das Sichtbarkeits-Timeout bestimmter Nachrichten in einer Warteschlange auf neue Werte, indem Sie [ChangeMessageVisibilityBatch](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called "Authentifizierung mit AWS"](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called "Installation"](#).

Ändern Sie das Sichtbarkeits-Timeout mehrerer Nachrichten

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Senden und Empfangen von Nachrichten in Amazon SQS mit AWS SDK für PHP Version 3

Weitere Informationen zu Amazon SQS SQS-Nachrichten finden Sie unter [Senden einer Nachricht an eine SQS-Warteschlange](#) und [Empfangen und Löschen einer Nachricht aus einer SQS-Warteschlange](#) im Service Quotas Quotas-Benutzerhandbuch.

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Senden Sie eine Nachricht an eine angegebene Warteschlange mit [SendMessage](#)
- Ruft eine oder mehrere Nachrichten (bis zu 10) aus einer angegebenen Warteschlange ab mit [ReceiveMessage](#).
- Löscht eine Nachricht aus einer Warteschlange mit [DeleteMessage](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter beschrieben [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter beschrieben [the section called “Installation”](#).

Senden einer Nachricht

Importe

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Sqs\SqsClient;
```

Beispiel-Code


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

Nachrichten empfangen und löschen

Importe

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

use Aws\Exception\AwsException;
```

```
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS mit Version 3 AWS SDK für PHP

Bei einer Warteschlange für unzustellbare Nachrichten handelt es sich um eine Warteschlange, an die andere (Quell-) Warteschlangen Nachrichten senden können, die nicht erfolgreich verarbeitet werden konnten. Sie können diese Nachrichten in der Warteschlange für unzustellbare Nachrichten

sammeln und isolieren, um festzustellen, warum die Verarbeitung fehlgeschlagen ist. Sie müssen jede Quellwarteschlange, die Nachrichten an eine Warteschlange für unzustellbare Nachrichten sendet, individuell konfigurieren. Eine Warteschlange für unzustellbare Nachrichten kann von mehreren Warteschlangen verwendet werden.

Weitere Informationen finden Sie unter [Verwenden von SQS-Warteschlangen für unzustellbare Nachrichten](#).

Das folgende Beispiel zeigt eine Anleitung für:

- Aktivieren Sie eine Warteschlange für unzustellbare Briefe mit. [SetQueueAttributes](#)

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar](#). [GitHub](#)

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben](#) [the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben](#) [the section called “Installation”](#).

Aktivieren Sie eine Warteschlange für unzustellbare Briefe

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Verwenden von Warteschlangen in Amazon SQS mit Version 3 AWS SDK für PHP

Weitere Informationen zu Amazon SQS SQS-Warteschlangen finden Sie unter [So funktionieren SQS-Warteschlangen](#).

In den nachstehenden Beispielen wird Folgendes veranschaulicht:

- Geben Sie eine Liste Ihrer Warteschlangen zurück mit [ListQueues](#)
- Erstellen Sie eine neue Warteschlange mit [CreateQueue](#).
- Gibt die URL einer vorhandenen Warteschlange zurück mit [GetQueueUrl](#).
- Löscht eine angegebene Warteschlange mit [DeleteQueue](#).

Der gesamte Beispielcode für AWS SDK für PHP ist [hier verfügbar GitHub](#).

Anmeldeinformationen

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen wie unter [beschrieben the section called “Authentifizierung mit AWS”](#). Importieren Sie dann die AWS SDK für PHP, wie unter [beschrieben the section called “Installation”](#).

Gibt eine Liste von Warteschlangen zurück

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Erstellen einer Warteschlange

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Gibt die URL einer Warteschlange zurück

Importe

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

use Aws\Exception\AwsException;
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Löschen einer Warteschlange

Importe

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Sqs\SqsClient;
```

Beispiel-Code

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

Senden Sie Ereignisse mit der AWS SDK für PHP Version 3 an EventBridge globale Amazon-Endpunkte

Sie können die [EventBridge globalen Endpunkte von Amazon](#) verwenden, um die Verfügbarkeit und Zuverlässigkeit Ihrer ereignisgesteuerten Anwendungen zu verbessern.

Nachdem der EventBridge globale Endpunkt [eingerichtet](#) ist, können Sie mithilfe des SDK for PHP Ereignisse an ihn senden.

 Important

Um EventBridge globale Endpunkte mit dem SDK for PHP zu verwenden, muss in Ihrer PHP-Umgebung die [AWS Common Runtime \(AWS CRT\) -Erweiterung](#) installiert sein.

Das folgende Beispiel verwendet die [PutEvents](#) Methode von, `EventBridgeClient` um ein einzelnes Ereignis an einen EventBridge globalen Endpunkt zu senden.

```
<?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]
]);
```

[Dieser Blogbeitrag](#) enthält weitere Informationen zu EventBridge globalen Endpunkten.

SDK for PHP PHP-Codebeispiele

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie AWS SDK für PHP with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielskategorien, die zeigen, wie Bibliotheken oder dienstspezifische Funktionen genutzt werden können.

Dienstleistungen

- [API-Gateway-Beispiele unter Verwendung von SDK für PHP](#)
- [Aurora-Beispiele unter Verwendung von SDK für PHP](#)
- [Auto-Scaling-Beispiele unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon Bedrock unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon DocumentDB unter Verwendung von SDK für PHP](#)
- [DynamoDB-Beispiele unter Verwendung von SDK für PHP](#)
- [EC2 Amazon-Beispiele mit SDK for PHP](#)
- [AWS Glue Beispiele mit SDK for PHP](#)
- [IAM-Beispiele unter Verwendung von SDK für PHP](#)
- [Kinesis-Beispiele unter Verwendung von SDK für PHP](#)
- [AWS KMS Beispiele mit SDK for PHP](#)
- [Lambda-Beispiele unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon MSK unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon RDS unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon RDS Data Service unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon Rekognition unter Verwendung von SDK für PHP](#)

- [Beispiele für Amazon S3 unter Verwendung von SDK für PHP](#)
- [Beispiele für S3 Directory Buckets unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon SES unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon SNS unter Verwendung von SDK für PHP](#)
- [Beispiele für Amazon SQS unter Verwendung von SDK für PHP](#)

API-Gateway-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für PHP with API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Aktionen](#)
- [Szenarien](#)

Aktionen

GetBasePathMapping

Das folgende Codebeispiel zeigt, wie Sie `GetBasePathMapping`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.
// getTheBasePathMapping();

```

- Einzelheiten zur API finden Sie [GetBasePathMapping](#) in der AWS SDK für PHP API-Referenz.

ListBasePathMappings

Das folgende Codebeispiel zeigt die Verwendung `ListBasePathMappings`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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();

```

- Einzelheiten zur API finden Sie [ListBasePathMappings](#) in der AWS SDK für PHP API-Referenz.

UpdateBasePathMapping

Das folgende Codebeispiel zeigt die Verwendung `UpdateBasePathMapping`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
```

- Einzelheiten zur API finden Sie [UpdateBasePathMapping](#) in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Aurora-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Szenarien](#)

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein mit React.js erstelltes Frontend verwendet, um mit einem RESTful PHP-Backend zu interagieren.

- Integrieren Sie eine React.js -Webanwendung in AWS Dienste.
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Erste Schritte](#)
- [Grundlagen](#)
- [Aktionen](#)

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen können.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon EC2 Auto Scaling Scaling-Gruppe mit einer Startvorlage und Availability Zones und erhalten Sie Informationen über laufende Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann Ressourcen.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);  
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Aktionen

CreateAutoScalingGroup

Das folgende Codebeispiel zeigt, wie man es benutztCreateAutoScalingGroup.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Einzelheiten zur API finden Sie [CreateAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

DeleteAutoScalingGroup

Das folgende Codebeispiel zeigt die Verwendung `DeleteAutoScalingGroup`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function deleteAutoScalingGroup($autoScalingGroupName)
{
    return $this->autoScalingClient->deleteAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'ForceDelete' => true,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

DescribeAutoScalingGroups

Das folgende Codebeispiel zeigt die Verwendung `DescribeAutoScalingGroups`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingGroups](#) in der AWS SDK für PHP API-Referenz.

DescribeAutoScalingInstances

Das folgende Codebeispiel zeigt die Verwendung `DescribeAutoScalingInstances`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeAutoScalingInstances](#) in der AWS SDK für PHP API-Referenz.

DescribeScalingActivities

Das folgende Codebeispiel zeigt die Verwendung `DescribeScalingActivities`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DescribeScalingActivities](#) in der AWS SDK für PHP API-Referenz.

DisableMetricsCollection

Das folgende Codebeispiel zeigt die Verwendung `DisableMetricsCollection`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DisableMetricsCollection](#) in der AWS SDK für PHP API-Referenz.

EnableMetricsCollection

Das folgende Codebeispiel zeigt die Verwendung `EnableMetricsCollection`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
```

```
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
}
```

- Einzelheiten zur API finden Sie [EnableMetricsCollection](#) in der AWS SDK für PHP API-Referenz.

SetDesiredCapacity

Das folgende Codebeispiel zeigt die Verwendung `SetDesiredCapacity`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'DesiredCapacity' => $desiredCapacity,
    ]);
}
```

- Einzelheiten zur API finden Sie [SetDesiredCapacity](#) in der AWS SDK für PHP API-Referenz.

TerminateInstanceInAutoScalingGroup

Das folgende Codebeispiel zeigt die Verwendung `TerminateInstanceInAutoScalingGroup`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
        }
    }
}
```

- Einzelheiten zur API finden Sie [TerminateInstanceInAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

UpdateAutoScalingGroup

Das folgende Codebeispiel zeigt die Verwendung `UpdateAutoScalingGroup`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
}
```

- Einzelheiten zur API finden Sie [UpdateAutoScalingGroup](#) in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon Bedrock unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Aktionen](#)

Aktionen

ListFoundationModels

Das folgende Codebeispiel zeigt, wie Sie `ListFoundationModels`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- Einzelheiten zur API finden Sie [ListFoundationModels](#) in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Szenarien](#)
- [Amazon Nova](#)
- [Amazon Titan Image Generator](#)
- [Anthropic Claude](#)
- [Stabile Diffusion](#)

Szenarien

Aufrufen mehrerer Basismodelle in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie eine Aufforderung vorbereiten und an eine Vielzahl von großsprachigen Modellen (LLMs) auf Amazon Bedrock senden

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie mehrere LLMs auf Amazon Bedrock auf.

```
namespace BedrockRuntime;
```

```

class GettingStartedWithBedrockRuntime
{
    protected BedrockRuntimeService $bedrockRuntimeService;
    public function runExample()
    {
        echo "\n";
        echo "-----\n";
        echo "Welcome to the Amazon Bedrock Runtime getting started demo using PHP!\n";
        echo "-----\n";
        $bedrockRuntimeService = new BedrockRuntimeService();
        $prompt = 'In one paragraph, who are you?';
        echo "\nPrompt: " . $prompt;
        echo "\n\nAnthropic Claude:\n";
        echo $bedrockRuntimeService->invokeClaude($prompt);
        echo
        "\n-----\n";
        $image_prompt = 'stylized picture of a cute old steampunk robot';
        echo "\nImage prompt: " . $image_prompt;
        echo "\n\nStability.ai Stable Diffusion XL:\n";
        $diffusionSeed = rand(0, 4294967295);
        $style_preset = 'photographic';
        $base64 = $bedrockRuntimeService->invokeStableDiffusion($image_prompt,
        $diffusionSeed, $style_preset);
        $image_path = $this->saveImage($base64, 'stability.stable-diffusion-xl');
        echo "The generated image has been saved to $image_path";
        echo "\n\nAmazon Titan Image Generation:\n";
        $titanSeed = rand(0, 2147483647);
        $base64 = $bedrockRuntimeService->invokeTitanImage($image_prompt,
        $titanSeed);
        $image_path = $this->saveImage($base64, 'amazon.titan-image-generator-v2');
        echo "The generated image has been saved to $image_path";
    }

    private function saveImage($base64_image_data, $model_id): string
    {
        $output_dir = "output";
        if (!file_exists($output_dir)) {
            mkdir($output_dir);
        }

        $i = 1;
    }
}

```

```
        while (file_exists("$output_dir/$model_id" . '_' . "$i.png")) {
            $i++;
        }

        $image_data = base64_decode($base64_image_data);
        $file_path = "$output_dir/$model_id" . '_' . "$i.png";
        $file = fopen($file_path, 'wb');
        fwrite($file, $image_data);
        fclose($file);
        return $file_path;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [InvokeModel](#)
 - [InvokeModelWithResponseStream](#)

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// 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();
```

- Weitere API-Informationen finden Sie unter [Converse](#) in der AWS SDK für PHP -API-Referenz.

Amazon Titan Image Generator

InvokeModel

Die folgenden Codebeispiele zeigen, wie Sie Amazon Titan Image in Amazon Bedrock aufrufen, um ein Bild zu generieren.

SDK für PHP

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
public function invokeTitanImage(string $prompt, int $seed)  
{  
    // The different model providers have individual request and response  
    formats.  
    // For the format, ranges, and default values for Titan Image models refer  
    to:  
    // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-  
    titan-image.html  
  
    $base64_image_data = "";  
    try {  
        $modelId = 'amazon.titan-image-generator-v2:0';  
        $request = json_encode([
```

```

        'taskType' => 'TEXT_IMAGE',
        'textToImageParams' => [
            'text' => $prompt
        ],
        'imageGenerationConfig' => [
            'numberOfImages' => 1,
            'quality' => 'standard',
            'cfgScale' => 8.0,
            'height' => 512,
            'width' => 512,
            'seed' => $seed
        ]
    ]);
    $result = $this->bedrockRuntimeClient->invokeModel([
        'contentType' => 'application/json',
        'body' => $request,
        'modelId' => $modelId,
    ]);
    $response_body = json_decode($result['body']);
    $base64_image_data = $response_body->images[0];
} catch (Exception $e) {
    echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
}

return $base64_image_data;
}

```

- Einzelheiten zur API finden Sie [InvokeModel](#) in der AWS SDK für PHP API-Referenz.

Anthropic Claude

InvokeModel

Das folgende Codebeispiel zeigt, wie mit der Invoke-Model-API eine Textnachricht an Anthropic Claude gesendet wird.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie das Basismodell von Anthropic Claude 2 auf, um Text zu generieren.

```
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;
    }
```

- Einzelheiten zur API finden Sie [InvokeModel](#) in der AWS SDK für PHP API-Referenz.

Stabile Diffusion

InvokeModel

Das folgende Codebeispiel zeigt, wie Sie Stability.ai Stable Diffusion XL in Amazon Bedrock aufrufen, um ein Bild zu generieren.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie ein Bild mit 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;
}
```

- Einzelheiten zur API finden Sie [InvokeModel](#) in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon DocumentDB unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren können.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Serverless-Beispiele](#)

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von 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();
```

DynamoDB-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Grundlagen](#)
- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Da in diesem Beispiel Hilfsdateien verwendet werden, [lesen Sie die Anleitungen](#) in der README.md-Datei für die PHP-Beispiele.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaller;
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);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Aktionen

BatchExecuteStatement

Das folgende Codebeispiel zeigt, wie man es benutzt `BatchExecuteStatement`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [BatchExecuteStatement](#) in der AWS SDK für PHP API-Referenz.

BatchWriteItem

Das folgende Codebeispiel zeigt die Verwendung `BatchWriteItem`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
            }
        }
    }
}

```

- Einzelheiten zur API finden Sie [BatchWriteItem](#) in der AWS SDK für PHP API-Referenz.

CreateTable

Das folgende Codebeispiel zeigt die Verwendung `CreateTable`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie eine -Tabelle.

```

$tableName = "ddb_demo_table_$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],
    ]);
}

```

- Einzelheiten zur API finden Sie [CreateTable](#) in der AWS SDK für PHP API-Referenz.

DeleteItem

Das folgende Codebeispiel zeigt die Verwendung `DeleteItem`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteItem](#) in der AWS SDK für PHP API-Referenz.

DeleteTable

Das folgende Codebeispiel zeigt die Verwendung `DeleteTable`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- Einzelheiten zur API finden Sie [DeleteTable](#) in der AWS SDK für PHP API-Referenz.

ExecuteStatement

Das folgende Codebeispiel zeigt die Verwendung `ExecuteStatement`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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,
    ]);
}
```

- Einzelheiten zur API finden Sie [ExecuteStatement](#) in der AWS SDK für PHP API-Referenz.

GetItem

Das folgende Codebeispiel zeigt die Verwendung `GetItem`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
    ]);
}
```

- Einzelheiten zur API finden Sie [GetItem](#) in der AWS SDK für PHP API-Referenz.

ListTables

Das folgende Codebeispiel zeigt die Verwendung `ListTables`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- Einzelheiten zur API finden Sie [ListTables](#) in der AWS SDK für PHP API-Referenz.

PutItem

Das folgende Codebeispiel zeigt die Verwendung `PutItem`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
}
```

- Einzelheiten zur API finden Sie [PutItem](#) in der AWS SDK für PHP API-Referenz.

Query

Das folgende Codebeispiel zeigt die Verwendung Query.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$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);
}
```

- Weitere API-Informationen finden Sie unter [Abfragen](#) in der AWS SDK für PHP -API-Referenz.

Scan

Das folgende Codebeispiel zeigt, wie man es benutzt.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
}

```

- Weitere API-Informationen finden Sie unter [Scan](#) in der AWS SDK für PHP -API-Referenz.

UpdateItem

Das folgende Codebeispiel zeigt, wie man es benutzt `UpdateItem`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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  
    ]  
],  
]);  
}
```

- Einzelheiten zur API finden Sie [UpdateItem](#) in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaller;
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,
            ],
        ],
    ],
]);
}

```

- Einzelheiten zur API finden Sie [BatchExecuteStatement](#) in der AWS SDK für PHP API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaller;

```

```

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,
    ]);
}
```

- Einzelheiten zur API finden Sie [ExecuteStatement](#) in der AWS SDK für PHP API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von 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);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von 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 Amazon-Beispiele mit SDK for PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für PHP mit Amazon Aktionen ausführen und allgemeine Szenarien implementieren EC2.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Aktionen](#)

Aktionen

CreateVpc

Das folgende Codebeispiel zeigt, wie Sie `CreateVpc`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
}
}

```

- Einzelheiten zur API finden Sie [CreateVpc](#) in der AWS SDK für PHP API-Referenz.

CreateVpcEndpoint

Das folgende Codebeispiel zeigt die Verwendung `CreateVpcEndpoint`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [CreateVpcEndpoint](#) in der AWS SDK für PHP API-Referenz.

DeleteVpc

Das folgende Codebeispiel zeigt die Verwendung `DeleteVpc`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```


- Einzelheiten zur API finden Sie [DeleteVpc](#) in der AWS SDK für PHP API-Referenz.

DeleteVpcEndpoints

Das folgende Codebeispiel zeigt die Verwendung `DeleteVpcEndpoints`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteVpcEndpoints](#) in der AWS SDK für PHP API-Referenz.

DescribeRouteTables

Das folgende Codebeispiel zeigt die Verwendung `DescribeRouteTables`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
}
```

- Einzelheiten zur API finden Sie [DescribeRouteTables](#) in der AWS SDK für PHP API-Referenz.

AWS Glue Beispiele mit SDK for PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für PHP with Aktionen ausführen und allgemeine Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Grundlagen](#)
- [Aktionen](#)

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon-S3-Bucket crawlt und eine Datenbank mit CSV-formatierten Metadaten generiert.
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf AWS Glue Data Catalog.
- Erstellen Sie einen Auftrag, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten umzuwandeln und die JSON-formatierte Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter [Tutorial: Erste Schritte mit AWS Glue Studio](#).

SDK für PHP

 Note

Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)

- [StartJobRun](#)

Aktionen

CreateCrawler

Das folgende Codebeispiel zeigt, wie man es benutzt `CreateCrawler`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
                    ]]
            ],
        ]);
    });
}
```

- Einzelheiten zur API finden Sie [CreateCrawler](#) in der AWS SDK für PHP API-Referenz.

CreateJob

Das folgende Codebeispiel zeigt die Verwendung `CreateJob`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
    ]);
}
```

- Einzelheiten zur API finden Sie [CreateJob](#) in der AWS SDK für PHP API-Referenz.

DeleteCrawler

Das folgende Codebeispiel zeigt die Verwendung `DeleteCrawler`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);

public function deleteCrawler($crawlerName)
{
    return $this->glueClient->deleteCrawler([
        'Name' => $crawlerName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteCrawler](#) in der AWS SDK für PHP API-Referenz.

DeleteDatabase

Das folgende Codebeispiel zeigt die Verwendung `DeleteDatabase`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
echo "Delete the databases.\n";
```

```
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

public function deleteDatabase($databaseName)
{
    return $this->glueClient->deleteDatabase([
        'Name' => $databaseName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteDatabase](#) in der AWS SDK für PHP API-Referenz.

DeleteJob

Das folgende Codebeispiel zeigt die Verwendung `DeleteJob`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

public function deleteJob($jobName)
{
    return $this->glueClient->deleteJob([
        'JobName' => $jobName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteJob](#) in der AWS SDK für PHP API-Referenz.

DeleteTable

Das folgende Codebeispiel zeigt die Verwendung `DeleteTable`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteTable](#) in der AWS SDK für PHP API-Referenz.

GetCrawler

Das folgende Codebeispiel zeigt die Verwendung `GetCrawler`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```

        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,
            ]);
        });
    }
}

```

- Einzelheiten zur API finden Sie [GetCrawler](#) in der AWS SDK für PHP API-Referenz.

GetDatabase

Das folgende Codebeispiel zeigt die Verwendung `GetDatabase`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

        $databaseName = "doc-example-database-$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,
    ]);
});
}

```

- Einzelheiten zur API finden Sie [GetDatabase](#) in der AWS SDK für PHP API-Referenz.

GetJobRun

Das folgende Codebeispiel zeigt die Verwendung `GetJobRun`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

$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,

```

```
    });  
}
```

- Einzelheiten zur API finden Sie [GetJobRun](#)in der AWS SDK für PHP API-Referenz.

GetJobRuns

Das folgende Codebeispiel zeigt die Verwendung `GetJobRuns`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);  
}
```

- Einzelheiten zur API finden Sie [GetJobRuns](#)in der AWS SDK für PHP API-Referenz.

GetTables

Das folgende Codebeispiel zeigt die Verwendung `GetTables`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$databaseName = "doc-example-database-$uniqid";

$tables = $glueService->getTables($databaseName);

public function getTables($databaseName): Result
{
    return $this->glueClient->getTables([
        'DatabaseName' => $databaseName,
    ]);
}
```

- Einzelheiten zur API finden Sie [GetTables](#) in der AWS SDK für PHP API-Referenz.

ListJobs

Das folgende Codebeispiel zeigt die Verwendung `ListJobs`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
}
```

- Einzelheiten zur API finden Sie [ListJobs](#) in der AWS SDK für PHP API-Referenz.

StartCrawler

Das folgende Codebeispiel zeigt die Verwendung `StartCrawler`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$crawlerName = "example-crawler-test-" . $uniqid;

$databaseName = "doc-example-database-$uniqid";

$glueService->startCrawler($crawlerName);

public function startCrawler($crawlerName): Result
{
    return $this->glueClient->startCrawler([
        'Name' => $crawlerName,
    ]);
}
```

```
}
```

- Einzelheiten zur API finden Sie [StartCrawler](#) in der AWS SDK für PHP API-Referenz.

StartJobRun

Das folgende Codebeispiel zeigt die Verwendung `StartJobRun`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
        ],
    ]);
}
```

- Einzelheiten zur API finden Sie [StartJobRun](#) in der AWS SDK für PHP API-Referenz.

IAM-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für PHP mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Grundlagen](#)
- [Aktionen](#)

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

Warning

Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie [AWS IAM Identity Center](#).

- Erstellen Sie einen Benutzer ohne Berechtigungen.

- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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";

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)

- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Aktionen

AttachRolePolicy

Das folgende Codebeispiel zeigt, wie man es benutzt `AttachRolePolicy`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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,
            ]);
        });
    }
}

```

- Einzelheiten zur API finden Sie [AttachRolePolicy](#) in der AWS SDK für PHP API-Referenz.

CreatePolicy

Das folgende Codebeispiel zeigt die Verwendung `CreatePolicy`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```

$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'];
}

```

- Einzelheiten zur API finden Sie [CreatePolicy](#) in der AWS SDK für PHP API-Referenz.

CreateRole

Das folgende Codebeispiel zeigt die Verwendung `CreateRole`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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'];
}

```

- Einzelheiten zur API finden Sie [CreateRole](#) in der AWS SDK für PHP API-Referenz.

CreateServiceLinkedRole

Das folgende Codebeispiel zeigt die Verwendung `CreateServiceLinkedRole`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
    }
```

- Einzelheiten zur API finden Sie [CreateServiceLinkedRole](#) in der AWS SDK für PHP API-Referenz.

CreateUser

Das folgende Codebeispiel zeigt die Verwendung `CreateUser`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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'];
}
```

- Einzelheiten zur API finden Sie [CreateUser](#) in der AWS SDK für PHP API-Referenz.

GetAccountPasswordPolicy

Das folgende Codebeispiel zeigt die Verwendung `GetAccountPasswordPolicy`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

public function getAccountPasswordPolicy()
{
```



```
        return $this->iamClient->getAccountPasswordPolicy();  
    }
```

- Einzelheiten zur API finden Sie [GetAccountPasswordPolicy](#) in der AWS SDK für PHP API-Referenz.

GetPolicy

Das folgende Codebeispiel zeigt die Verwendung `GetPolicy`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$uuid = uniqid();  
$service = new IAMService();  
  
public function getPolicy($policyArn)  
{  
    return $this->customWaiter(function () use ($policyArn) {  
        return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);  
    });  
}
```

- Einzelheiten zur API finden Sie [GetPolicy](#) in der AWS SDK für PHP API-Referenz.

GetRole

Das folgende Codebeispiel zeigt die Verwendung `GetRole`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

public function getRole($roleName)
{
    return $this->customWaiter(function () use ($roleName) {
        return $this->iamClient->getRole(['RoleName' => $roleName]);
    });
}
```

- Einzelheiten zur API finden Sie [GetRole](#) in der AWS SDK für PHP API-Referenz.

ListAttachedRolePolicies

Das folgende Codebeispiel zeigt die Verwendung `ListAttachedRolePolicies`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
    }

```

- Einzelheiten zur API finden Sie [ListAttachedRolePolicies](#) in der AWS SDK für PHP API-Referenz.

ListGroups

Das folgende Codebeispiel zeigt die Verwendung `ListGroups`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

$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);
    }
}
```

- Einzelheiten zur API finden Sie [ListGroups](#) in der AWS SDK für PHP API-Referenz.

ListPolicies

Das folgende Codebeispiel zeigt die Verwendung `ListPolicies`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
}
```

- Einzelheiten zur API finden Sie [ListPolicies](#) in der AWS SDK für PHP API-Referenz.

ListRolePolicies

Das folgende Codebeispiel zeigt die Verwendung `ListRolePolicies`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [ListRolePolicies](#) in der AWS SDK für PHP API-Referenz.

ListRoles

Das folgende Codebeispiel zeigt die Verwendung `ListRoles`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);
}
```

- Einzelheiten zur API finden Sie [ListRoles](#) in der AWS SDK für PHP API-Referenz.

ListSAMLProviders

Das folgende Codebeispiel zeigt die Verwendung `ListSAMLProviders`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

public function listSAMLProviders()
{
    return $this->iamClient->listSAMLProviders();
}
```

- Einzelheiten zur API finden Sie unter [Liste SAMLProviders](#) in der AWS SDK für PHP API-Referenz.

ListUsers

Das folgende Codebeispiel zeigt die Verwendung `ListUsers`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
$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);  
}
```

- Einzelheiten zur API finden Sie [ListUsers](#) in der AWS SDK für PHP API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Kinesis Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Serverless-Beispiele](#)

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von 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);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von 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 Beispiele mit SDK for PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für PHP with Aktionen ausführen und allgemeine Szenarien implementieren AWS KMS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Erste Schritte](#)
- [Grundlagen](#)
- [Aktionen](#)

Erste Schritte

Hallo AWS KMS

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie [ListKeys](#) in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines KMS-Schlüssels.
- Listen Sie die KMS-Schlüssel für Ihr Konto auf und rufen Sie Details über sie ab.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.
- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Taggen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CreateAlias](#)
 - [CreateGrant](#)
 - [CreateKey](#)
 - [Decrypt](#)
 - [DescribeKey](#)
 - [DisableKey](#)
 - [EnableKey](#)
 - [Encrypt](#)
 - [GetKeyPolicy](#)
 - [ListAliases](#)
 - [ListGrants](#)
 - [ListKeys](#)
 - [RevokeGrant](#)
 - [ScheduleKeyDeletion](#)
 - [Sign](#)
 - [TagResource](#)

Aktionen

CreateAlias

Das folgende Codebeispiel zeigt, wie man es benutzt `CreateAlias`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```

    /**
     * @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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [CreateAlias](#) in der AWS SDK für PHP API-Referenz.

CreateGrant

Das folgende Codebeispiel zeigt die Verwendung `CreateGrant`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    /**
     * @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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [CreateGrant](#) in der AWS SDK für PHP API-Referenz.

CreateKey

Das folgende Codebeispiel zeigt die Verwendung `CreateKey`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    /**
     * @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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [CreateKey](#) in der AWS SDK für PHP API-Referenz.

Decrypt

Das folgende Codebeispiel zeigt die Verwendung `Decrypt`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Weitere API-Informationen finden Sie unter [Decrypt](#) in der AWS SDK für PHP -API-Referenz.

DeleteAlias

Das folgende Codebeispiel zeigt, wie man es benutztDeleteAlias.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteAlias](#) in der AWS SDK für PHP API-Referenz.

DescribeKey

Das folgende Codebeispiel zeigt die Verwendung `DescribeKey`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [DescribeKey](#) in der AWS SDK für PHP API-Referenz.

DisableKey

Das folgende Codebeispiel zeigt die Verwendung `DisableKey`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [DisableKey](#) in der AWS SDK für PHP API-Referenz.

EnableKey

Das folgende Codebeispiel zeigt die Verwendung `EnableKey`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * @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;
    }
}

```

- Einzelheiten zur API finden Sie [EnableKey](#) in der AWS SDK für PHP API-Referenz.

Encrypt

Das folgende Codebeispiel zeigt die Verwendung `Encrypt`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * @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;
    }
}

```

- Weitere API-Informationen finden Sie unter [Encrypt](#) in der AWS SDK für PHP -API-Referenz.

ListAliases

Das folgende Codebeispiel zeigt, wie man es benutzt `ListAliases`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * @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;  
    }  
}
```

- Einzelheiten zur API finden Sie [ListAliases](#) in der AWS SDK für PHP API-Referenz.

ListGrants

Das folgende Codebeispiel zeigt die Verwendung `ListGrants`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**  
 * @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;  
    }  
}
```

- Einzelheiten zur API finden Sie [ListGrants](#) in der AWS SDK für PHP API-Referenz.

ListKeys

Das folgende Codebeispiel zeigt die Verwendung `ListKeys`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [ListKeys](#) in der AWS SDK für PHP API-Referenz.

PutKeyPolicy

Das folgende Codebeispiel zeigt die Verwendung `PutKeyPolicy`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [PutKeyPolicy](#) in der AWS SDK für PHP API-Referenz.

RevokeGrant

Das folgende Codebeispiel zeigt die Verwendung `RevokeGrant`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [RevokeGrant](#) in der AWS SDK für PHP API-Referenz.

ScheduleKeyDeletion

Das folgende Codebeispiel zeigt die Verwendung `ScheduleKeyDeletion`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [ScheduleKeyDeletion](#) in der AWS SDK für PHP API-Referenz.

Sign

Das folgende Codebeispiel zeigt die Verwendung `Sign`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Weitere API-Informationen finden Sie unter [Sign](#) in der AWS SDK für PHP -API-Referenz.

TagResource

Das folgende Codebeispiel zeigt, wie man es benutztTagResource.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * @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;
    }
}
```

- Einzelheiten zur API finden Sie [TagResource](#) in der AWS SDK für PHP API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Grundlagen](#)
- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter [Erstellen einer Lambda-Funktion mit der Konsole](#).

SDK für PHP

 Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use IAM\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("-----\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("-----\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
```

```

        \ "Service\: \"lambda.amazonaws.com\"
    },
    \ "Action\: \"sts:AssumeRole\"
}
]
}";
$role = $iamService->createRole($roleName, $rolePolicyDocument);
echo "Created role {$role['RoleName']}.\\n";

$iamService->attachRolePolicy(
    $role['RoleName'],
    "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
);
echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\\n";

echo "\\nNow let's create an S3 bucket and upload our Lambda code there.\\n";
$bucketName = "amzn-s3-demo-bucket-$uniqid";
$s3client->createBucket([
    'Bucket' => $bucketName,
]);
echo "Created bucket $bucketName.\\n";

$functionName = "doc_example_lambda_$uniqid";
$codeBasic = __DIR__ . "/lambda_handler_basic.zip";
$handler = "lambda_handler_basic";
$file = file_get_contents($codeBasic);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => $functionName,
    'Body' => $file,
]);
echo "Uploaded the Lambda code.\\n";

$createLambdaFunction = $lambdaService->createFunction($functionName, $role,
$bucketName, $handler);
// Wait until the function has finished being created.
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
    } while ($getLambdaFunction['Configuration']['State'] == "Pending");
    echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\\n";

    sleep(1);

```

```

    echo "\nOk, let's invoke that Lambda code.\n";
    $basicParams = [
        'action' => 'increment',
        'number' => 3,
    ];
    /** @var Stream $invokeFunction */
    $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
    $result = json_decode($invokeFunction->getContents())->result;
    echo "After invoking the Lambda code with the input of
    {$basicParams['number']} we received $result.\n";

    echo "\nSince that's working, let's update the Lambda code.\n";
    $codeCalculator = "lambda_handler_calculator.zip";
    $handlerCalculator = "lambda_handler_calculator";
    echo "First, put the new code into the S3 bucket.\n";
    $file = file_get_contents($codeCalculator);
    $s3client->putObject([
        'Bucket' => $bucketName,
        'Key' => $functionName,
        'Body' => $file,
    ]);
    echo "New code uploaded.\n";

    $lambdaService->updateFunctionCode($functionName, $bucketName,
    $functionName);
    // Wait for the Lambda code to finish updating.
    do {
        $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
    "Successful");
    echo "New Lambda code uploaded.\n";

    $environment = [
        'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
    ];
    $lambdaService->updateFunctionConfiguration($functionName,
    $handlerCalculator, $environment);
    do {
        $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);

```

```

    } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
    echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
information.\n";

    echo "Invoke the new code with some new data.\n";
    $calculatorParams = [
        'action' => 'plus',
        'x' => 5,
        'y' => 4,
    ];
    $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
    $result = json_decode($invokeFunction['Payload']->getContents())->result;
    echo "Indeed, {"$calculatorParams['x']} + {"$calculatorParams['y']} does equal
$result.\n";
    echo "Here's the extra debug info: ";
    echo base64_decode($invokeFunction['LogResult']) . "\n";

    echo "\nBut what happens if you try to divide by zero?\n";
    $divZeroParams = [
        'action' => 'divide',
        'x' => 5,
        'y' => 0,
    ];
    $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
    $result = json_decode($invokeFunction['Payload']->getContents())->result;
    echo "You get a |$result| result.\n";
    echo "And an error message: ";
    echo base64_decode($invokeFunction['LogResult']) . "\n";

    echo "\nHere's all the Lambda functions you have in this Region:\n";
    $listLambdaFunctions = $lambdaService->listFunctions(5);
    $allLambdaFunctions = $listLambdaFunctions['Functions'];
    $next = $listLambdaFunctions->get('NextMarker');
    while ($next != false) {
        $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
        $next = $listLambdaFunctions->get('NextMarker');
        $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
    }
    foreach ($allLambdaFunctions as $function) {
        echo "{$function['FunctionName']}\n";
    }

```

```

    }

    echo "\n\nAnd don't forget to clean up your data!\n";

    $lambdaService->deleteFunction($functionName);
    echo "Deleted Lambda function.\n";
    $iamService->deleteRole($role['RoleName']);
    echo "Deleted Role.\n";
    $deleteObjects = $s3client->listObjectsV2([
        'Bucket' => $bucketName,
    ]);
    $deleteObjects = $s3client->deleteObjects([
        'Bucket' => $bucketName,
        'Delete' => [
            'Objects' => $deleteObjects['Contents'],
        ]
    ]);
    echo "Deleted all objects from the S3 bucket.\n";
    $s3client->deleteBucket(['Bucket' => $bucketName]);
    echo "Deleted the bucket.\n";
}
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Aktionen

CreateFunction

Das folgende Codebeispiel zeigt, wie man es benutzt `CreateFunction`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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",
        ]);
    });
}
```

- Einzelheiten zur API finden Sie [CreateFunction](#) in der AWS SDK für PHP API-Referenz.

DeleteFunction

Das folgende Codebeispiel zeigt die Verwendung `DeleteFunction`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function deleteFunction($functionName)
{
    return $this->lambdaClient->deleteFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- Einzelheiten zur API finden Sie [DeleteFunction](#) in der AWS SDK für PHP API-Referenz.

GetFunction

Das folgende Codebeispiel zeigt die Verwendung `GetFunction`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- Einzelheiten zur API finden Sie [GetFunction](#) in der AWS SDK für PHP API-Referenz.

Invoke

Das folgende Codebeispiel zeigt die Verwendung `Invoke`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- Weitere API-Informationen finden Sie unter [Invoke](#) in der AWS SDK für PHP -API-Referenz.

ListFunctions

Das folgende Codebeispiel zeigt, wie man es benutztListFunctions.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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,
]);
}
```

- Einzelheiten zur API finden Sie [ListFunctions](#) in der AWS SDK für PHP API-Referenz.

UpdateFunctionCode

Das folgende Codebeispiel zeigt die Verwendung `UpdateFunctionCode`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
{
    return $this->lambdaClient->updateFunctionCode([
        'FunctionName' => $functionName,
        'S3Bucket' => $s3Bucket,
        'S3Key' => $s3Key,
    ]);
}
```

- Einzelheiten zur API finden Sie [UpdateFunctionCode](#) in der AWS SDK für PHP API-Referenz.

UpdateFunctionConfiguration

Das folgende Codebeispiel zeigt die Verwendung `UpdateFunctionConfiguration`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
{
    return $this->lambdaClient->updateFunctionConfiguration([
        'FunctionName' => $functionName,
        'Handler' => "$handler.lambda_handler",
        'Environment' => $environment,
    ]);
}
```

- Einzelheiten zur API finden Sie [UpdateFunctionConfiguration](#) in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit 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);

```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von 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);
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von 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);
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von 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();

```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von 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);
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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);
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

 Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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();
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von 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);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von 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);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von 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);

```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von 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);
```

Beispiele für Amazon MSK unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Serverless-Beispiele](#)

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von 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);

```

Beispiele für Amazon RDS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon RDS verwenden. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

Aktionen

CreateDBInstance

Das folgende Codebeispiel zeigt, wie Sie CreateDBInstance.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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";
}
```

- Einzelheiten zur API finden Sie unter DBInstance In der AWS SDK für PHP API-Referenz [erstellen](#).

CreateDBSnapshot

Das folgende Codebeispiel zeigt die Verwendung CreateDBSnapshot.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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";
}
```

- Einzelheiten zur API finden Sie unter DBSnapshot In der AWS SDK für PHP API-Referenz [erstellen](#).

DeleteDBInstance

Das folgende Codebeispiel zeigt die VerwendungDeleteDBInstance.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```
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";
}
```

- Einzelheiten zur API finden Sie unter [Löschen DBInstance](#) in der AWS SDK für PHP API-Referenz.

DescribeDBInstances

Das folgende Codebeispiel zeigt die Verwendung `DescribeDBInstances`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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";
}
```

- Einzelheiten zur API finden Sie unter [Describe DBInstances](#) in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-

Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein mit React.js erstelltes Frontend verwendet, um mit einem RESTful PHP-Backend zu interagieren.

- Integrieren Sie eine React.js -Webanwendung in AWS Dienste.
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit 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);
```

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Szenarien](#)

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein mit React.js erstelltes Frontend verwendet, um mit einem RESTful PHP-Backend zu interagieren.

- Integrieren Sie eine React.js -Webanwendung in AWS Dienste.
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.

- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Rekognition unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Szenarien](#)

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. [GitHub](#)

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Beispiele für Amazon S3 unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon S3 verwenden. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Erste Schritte](#)
- [Grundlagen](#)
- [Aktionen](#)

- [Szenarien](#)
- [Serverless-Beispiele](#)

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen können.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
use Aws\S3\S3Client;

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

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
```

```

    });
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {

```

```

        echo "Failed to list objects in $this->bucketName with error: " .
        $exception->getMessage();
        exit("Please fix error with listing objects before continuing.");
    }

    try {
        $objects = [];
        foreach ($contents['Contents'] as $content) {
            $objects[] = [
                'Key' => $content['Key'],
            ];
        }
        $this->s3client->deleteObjects([
            'Bucket' => $this->bucketName,
            'Delete' => [
                'Objects' => $objects,
            ],
        ]);
        $check = $this->s3client->listObjectsV2([
            'Bucket' => $this->bucketName,
        ]);
        if (isset($check['Contents']) && count($check['Contents']) > 0) {
            throw new Exception("Bucket wasn't empty.");
        }
        echo "Deleted all objects and folders from $this->bucketName.\n";
    } catch (Exception $exception) {
        echo "Failed to delete $fileName from $this->bucketName with error: " .
        $exception->getMessage();
        exit("Please fix error with object deletion before continuing.");
    }

    try {
        $this->s3client->deleteBucket([
            'Bucket' => $this->bucketName,
        ]);
        echo "Deleted bucket $this->bucketName.\n";
    } catch (Exception $exception) {
        echo "Failed to delete $this->bucketName with error: " . $exception-
        >getMessage();
        exit("Please fix error with bucket deletion before continuing.");
    }

    echo "Successfully ran the Amazon S3 with PHP demo.\n";

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Aktionen

CopyObject

Das folgende Codebeispiel zeigt, wie man es benutzt `CopyObject`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Einfache Kopie eines Objekts.

```
$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.");  
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für PHP API-Referenz.

CreateBucket

Das folgende Codebeispiel zeigt die Verwendung `CreateBucket`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen 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.");  
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für PHP API-Referenz.

DeleteBucket

Das folgende Codebeispiel zeigt die Verwendung `DeleteBucket`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie einen leeren Bucket.

```
$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.");
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für PHP API-Referenz.

DeleteObject

Das folgende Codebeispiel zeigt die Verwendung `DeleteObject`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für PHP API-Referenz.

DeleteObjects

Das folgende Codebeispiel zeigt die Verwendung `DeleteObjects`.

SDK für PHP

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie eine Reihe von Objekten aus einer Schlüsselliste.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für PHP API-Referenz.

GetObject

Das folgende Codebeispiel zeigt die Verwendung `GetObject`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie ein Objekt ab.

```

$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.");
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für PHP API-Referenz.

ListObjectsV2

Das folgende Codebeispiel zeigt die Verwendung `ListObjectsV2`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

Listen Sie Objekte in einem Bucket auf.

```

$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.");
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für PHP API-Referenz.

PutObject

Das folgende Codebeispiel zeigt die Verwendung `PutObject`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie ein Objekt in einen Bucket hoch.

```
$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.");
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für PHP API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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 \"\$preSignedUrl\" and 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;
    }
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft

den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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);
```

Beispiele für S3 Directory Buckets unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von S3 Directory Buckets Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Grundlagen](#)

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.
- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.
- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
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;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Beispiele für Amazon SES unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Szenarien](#)

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein mit React.js erstelltes Frontend verwendet, um mit einem RESTful PHP-Backend zu interagieren.

- Integrieren Sie eine React.js -Webanwendung in AWS Dienste.
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon SNS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Aktionen](#)
- [Szenarien](#)
- [Serverless-Beispiele](#)

Aktionen

CheckIfPhoneNumberIsOptedOut

Das folgende Codebeispiel zeigt, wie Sie `CheckIfPhoneNumberIsOptedOut`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [CheckIfPhoneNumbersOptedOut](#) in der AWS SDK für PHP API-Referenz.

ConfirmSubscription

Das folgende Codebeispiel zeigt die Verwendung `ConfirmSubscription`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie [ConfirmSubscription](#) in der AWS SDK für PHP API-Referenz.

CreateTopic

Das folgende Codebeispiel zeigt die Verwendung `CreateTopic`.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [CreateTopic](#) in der AWS SDK für PHP API-Referenz.

DeleteTopic

Das folgende Codebeispiel zeigt die Verwendung `DeleteTopic`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());  
}
```

- Einzelheiten zur API finden Sie [DeleteTopic](#) in der AWS SDK für PHP API-Referenz.

GetSMSAttributes

Das folgende Codebeispiel zeigt die Verwendung `GetSMSAttributes`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie unter [Get SMSAttributes](#) in AWS SDK für PHP API-Referenz.

GetTopicAttributes

Das folgende Codebeispiel zeigt die Verwendung `GetTopicAttributes`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetTopicAttributes](#) in der AWS SDK für PHP API-Referenz.

ListPhoneNumbersOptedOut

Das folgende Codebeispiel zeigt die Verwendung `ListPhoneNumbersOptedOut`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());  
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [ListPhoneNumbersOptedOut](#) in der AWS SDK für PHP API-Referenz.

ListSubscriptions

Das folgende Codebeispiel zeigt die Verwendung `ListSubscriptions`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie [ListSubscriptions](#) in der AWS SDK für PHP API-Referenz.

ListTopics

Das folgende Codebeispiel zeigt die Verwendung `ListTopics`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie [ListTopics](#) in der AWS SDK für PHP API-Referenz.

Publish

Das folgende Codebeispiel zeigt die Verwendung `Publish`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Weitere API-Informationen finden Sie unter [Publish](#) in der AWS SDK für PHP -API-Referenz.

SetSMSAttributes

Das folgende Codebeispiel zeigt, wie man es benutzt `SetSMSAttributes`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie unter [Set SMSAttributes](#) in der AWS SDK für PHP API-Referenz.

SetTopicAttributes

Das folgende Codebeispiel zeigt die Verwendung `SetTopicAttributes`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Einzelheiten zur API finden Sie [SetTopicAttributes](#) in der AWS SDK für PHP API-Referenz.

Subscribe

Das folgende Codebeispiel zeigt die Verwendung `Subscribe`.

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
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());
}
```

Abonnieren Sie einen HTTP-Endpunkt für ein Thema.

```
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());
}
```

- Weitere API-Informationen finden Sie unter [Subscribe](#) in der AWS SDK für PHP -API-Referenz.

Unsubscribe

Das folgende Codebeispiel zeigt, wie man es benutztUnsubscribe.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).

- Weitere API-Informationen finden Sie unter [Unsubscribe](#) in der AWS SDK für PHP -API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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());
}
```

- Weitere Informationen finden Sie im [AWS SDK für PHP -Entwicklerhandbuch](#).
- Weitere API-Informationen finden Sie unter [Publish](#) in der AWS SDK für PHP -API-Referenz.

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von 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();
```

Beispiele für Amazon SQS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- [Serverless-Beispiele](#)

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird.

Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von 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);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von 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);
```

Migration von Version 2 des AWS SDK für PHP

Dieses Thema zeigt, wie Sie Ihren Code auf die Version 3 des AWS SDK für PHP migrieren und wie sich die neue Version von der Version 2 des SDK unterscheidet.

Note

Das grundlegende Nutzungsmuster des SDK (d. h. `$result = $client->operation($params);`) hat sich von Version 2 auf Version 3 nicht geändert, was zu einer reibungslosen Migration führen sollte.

Einführung

Version 3 von AWS SDK für PHP stellt eine erhebliche Anstrengung dar, um die Funktionen des SDK zu verbessern, Kundenfeedback aus mehr als zwei Jahren einzubeziehen, unsere Abhängigkeiten zu erweitern, die Leistung zu verbessern und die neuesten PHP-Standards zu übernehmen.

Was ist neu in Version 3?

Version 3 von AWS SDK für PHP folgt den [Standards PSR-4 und PSR-7 und](#) wird auch in Zukunft dem [SemVer](#)Standard folgen.

Zu den weiteren neuen Funktionen gehören:

- Middleware-System zum Anpassen des Service-Client-Verhaltens
- Flexible Umbrüche zum Durchlaufen paginierter Ergebnisse
- Fähigkeit, Daten aus Ergebnis - und Paginator-Objekten abzufragen mit JMESPath
- Einfache Fehlersuche über die 'debug'-Konfigurationsoption

Entkoppelte HTTP-Schicht

- Standardmäßig wird [Guzzle 6](#) zum Senden von Anforderungen verwendet, aber Guzzle 5 wird ebenfalls unterstützt.
- Das SDK funktioniert in Umgebungen, in denen cURL nicht verfügbar ist.
- Benutzerdefinierte HTTP-Handler werden ebenfalls unterstützt.

Asynchrone Anfragen

- Funktionen wie Waiter und mehrteilige Uploads können ebenfalls asynchron verwendet werden.
- Asynchrone Workflows können mithilfe von Promises und Co-Routinen erstellt werden.
- Die Performance von gleichzeitigen oder im Stapel verarbeiteten Anfragen wurde verbessert.

Was ist anders als Version 2?

Die Projektabhängigkeiten werden aktualisiert

Die Abhängigkeiten des SDK wurden in dieser Version geändert.

- Das SDK benötigt jetzt PHP 8.1+. Wir verwenden [Generatoren](#) innerhalb des SDK-Code freier.
- Wir haben das SDK auf [Guzzle 6](#) (oder 5) aktualisiert, das die zugrunde liegende HTTP-Client-Implementierung bereitstellt, die vom SDK zum Senden von Anfragen an die Dienste verwendet wird. AWS Die neueste Version von Guzzle bringt eine Reihe von Verbesserungen mit sich, darunter asynchrone Anfragen, austauschbare HTTP-Handler, PSR-7-Konformität, bessere Performance und mehr.
- Das PSR-7-Paket aus PHP-FIG (`psr/http-message`) definiert Schnittstellen zur Darstellung von HTTP-Anfragen, HTTP-Antworten und Streams. URLs Diese Schnittstellen werden im gesamten SDK und Guzzle verwendet, was die Interoperabilität mit anderen PSR-7-kompatiblen Paketen gewährleistet.
- Die PSR-7-Implementierung von Guzzle (`guzzlehttp/psr7`) bietet eine Implementierung der Schnittstellen in PSR-7 und verschiedene hilfreiche Klassen und Funktionen. Sowohl das SDK als auch Guzzle 6 basieren weitgehend auf diesem Paket.
- Die [Promises/A+](#)-Implementierung von Guzzle (`guzzlehttp/promises`) wird im gesamten SDK und in Guzzle verwendet, um Schnittstellen für die Verwaltung von asynchronen Anforderungen und Co-Routinen bereitzustellen. Während der Multi-CURL-HTTP-Handler von Guzzle letztlich das nicht blockierende I/O Modell implementiert, das asynchrone Anfragen ermöglicht, bietet dieses Paket die Möglichkeit, innerhalb dieses Paradigmas zu programmieren. Weitere Informationen finden Sie unter [Versprechen in der Version 3. AWS SDK für PHP](#)
- Die PHP-Implementierung von [JMESPath](#) (`mt Dowling/jmespath.php`) wird im SDK verwendet, um die Datenabfragefähigkeit der `Aws\ResultPaginator::search()` Methoden `Aws\Result::search()` und bereitzustellen. Weitere Informationen finden Sie unter [JMESPath Ausdrücke in AWS SDK für PHP Version 3](#).

Regions- und Versionsoptionen sind jetzt erforderlich

Bei der Instanziierung eines Clients für beliebige Services müssen Sie die Optionen `'region'` und `'version'` angeben. In Version 2 von `'version'` war dies völlig optional und `'region'` manchmal optional. AWS SDK für PHP In Version 3 sind beide immer erforderlich. Wenn Sie beide Optionen explizit angeben, können Sie sich auf die API-Version und AWS Region festlegen, für die Sie programmieren. Wenn neue API-Versionen erstellt werden oder neue AWS Regionen verfügbar werden, sind Sie vor potenziell schwerwiegenden Änderungen geschützt, bis Sie bereit sind, Ihre Konfiguration explizit zu aktualisieren.

Note

Wenn es Ihnen nicht wichtig ist, welche API-Version Sie verwenden, können Sie einfach die Option `'version'` auf `'latest'` setzen. Allerdings empfehlen wir, dass Sie die API-Versionsnummer für Produktionscode explizit angeben.

Nicht alle Dienste sind in allen AWS Regionen verfügbar. Eine Liste der verfügbaren Regionen finden Sie [Regionen und Endpunkte](#).

Für Dienste, die nur über einen einzigen, globalen Endpunkt verfügbar sind (z. B. Amazon Route 53 und Amazon CloudFront) AWS Identity and Access Management, instanziiieren Sie Clients mit der konfigurierten Region auf `us-east-1`.

Important

Das SDK umfasst auch Clients mit mehreren Regionen, die Anfragen auf der Grundlage eines Parameters (`@region`), der als Befehlsparameter bereitgestellt wird, an verschiedene AWS Regionen senden können. Die von diesen Clients standardmäßig verwendete Region wird mit der Option `region` angegeben, die dem Client-Konstruktor übergeben wird.

Die Client-Instanziierung verwendet den Konstruktor

In Version 3 von hat sich die Art und Weise AWS SDK für PHP, wie Sie einen Client instanziiieren, geändert. Anstelle der `factory`-Methoden in Version 2, können Sie einen Client einfach mithilfe des Schlüsselworts `new` instanziiieren.

```
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

Das Instanzieren eines Clients mithilfe der `factory()`-Methode funktioniert weiterhin. Es gilt jedoch als veraltet.

Die Client-Konfiguration hat sich geändert

Die Client-Konfigurationsoptionen in Version 3 von AWS SDK für PHP haben sich gegenüber Version 2 geringfügig geändert. Eine Beschreibung aller unterstützten Optionen finden Sie auf der Seite [Konfiguration für AWS SDK für PHP Version 3](#).

Important

In Version 3 sind `'key'` und `'secret'` keine gültigen Optionen auf der Stammebene mehr, aber Sie können sie als Teil der `'credentials'`-Option übergeben. Ein Grund, warum wir das gemacht haben, war, Entwickler davon abzuhalten, ihre AWS Anmeldeinformationen fest in ihren Projekten zu programmieren.

Das Sdk-Objekt

Version 3 von AWS SDK für PHP führt das `Aws\Sdk` Objekt als Ersatz für ein `Aws\Common\Aws` Das Sdk-Objekt fungiert als Client-Factory und wird verwendet, um gemeinsame Konfigurationsoptionen für mehrere Clients zu verwalten.

Obwohl die `Aws`-Klasse in Version 2 des SDK funktioniert hat wie ein Service Locator (sie gab immer dieselbe Instance eines Clients zurück), gibt die `Sdk`-Klasse in Version 3 bei jeder Verwendung eine neue Instance eines Clients zurück.

Das Sdk-Objekt unterstützt auch nicht dasselbe Konfigurationsdateiformat wie Version 2 des SDK. Dieses Konfigurationsformat war spezifisch für Guzzle 3 und ist jetzt veraltet. Die Konfiguration kann einfacher mit grundlegenden Arrays erfolgen und ist in [Verwendung der Sdk-Klasse](#) dokumentiert.

Einige API-Ergebnisse haben sich geändert

Um Konsistenz bei der Analyse des Ergebnisses eines API-Vorgangs durch das SDK zu gewährleisten, verfügen Amazon ElastiCache, Amazon RDS und Amazon Redshift jetzt über ein zusätzliches Wrapping-Element für einige API-Antworten.

Zum Beispiel beinhaltet der Aufruf des Amazon [DescribeEngineDefaultParameters](#) RDS-Ergebnisses in Version 3 jetzt ein umschließendes „EngineDefaults“-Element. In Version 2 war dieses Element nicht vorhanden.

```
$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'];
```

Die folgenden Operationen sind betroffen und enthalten jetzt ein Wrapping-Element in der Ausgabe des Ergebnisses (nachfolgend in Klammern gezeigt):

- Amazon ElastiCache
 - AuthorizeCacheSecurityGroupIngress (CacheSecurityGroup)
 - CopySnapshot (Schnappschuss)
 - CreateCacheCluster (CacheCluster)
 - CreateCacheParameterGroup (CacheParameterGroup)
 - CreateCacheSecurityGroup (CacheSecurityGroup)
 - CreateCacheSubnetGroup (CacheSubnetGroup)

- CreateReplicationGroup (ReplicationGroup)
- CreateSnapshot (Schnappschuss)
- DeleteCacheCluster (CacheCluster)
- DeleteReplicationGroup (ReplicationGroup)
- DeleteSnapshot (Schnappschuss)
- DescribeEngineDefaultParameters (EngineDefaults)
- ModifyCacheCluster (CacheCluster)
- ModifyCacheSubnetGroup (CacheSubnetGroup)
- ModifyReplicationGroup (ReplicationGroup)
- PurchaseReservedCacheNodesOffering (ReservedCacheNode)
- RebootCacheCluster (CacheCluster)
- RevokeCacheSecurityGroupIngress (CacheSecurityGroup)
- Amazon RDS
 - AddSourceIdentifierToSubscription (EventSubscription)
 - Autorisieren DBSecurity GroupIngress (DBSecurityGruppe)
 - DBParameterGruppe (DBParameterGruppe) kopieren
 - Kopieren DBSnapshot (DBSnapshot)
 - CopyOptionGroup (OptionGroup)
 - Erstellen DBInstance (DBInstance)
 - Erstellen DBInstance ReadReplica (DBInstance)
 - DBParameterGruppe erstellen (DBParameterGruppe)
 - DBSecurityGruppe erstellen (DBSecurityGruppe)
 - Erstellen DBSnapshot (DBSnapshot)
 - DBSubnetGruppe erstellen (DBSubnetGruppe)
 - CreateEventSubscription (EventSubscription)
 - CreateOptionGroup (OptionGroup)
 - Löschen DBInstance (DBInstance)
 - Löschen DBSnapshot (DBSnapshot)
 - ~~DeleteEventSubscription (EventSubscription)~~
 - DescribeEngineDefaultParameters (EngineDefaults)

- Ändern DBInstance (DBInstance)
- DBSubnetGruppe ändern (DBSubnetGruppe)
- ModifyEventSubscription (EventSubscription)
- ModifyOptionGroup (OptionGroup)
- PromoteReadReplica (DBInstance)
- PurchaseReservedDBInstancesAngebot (ReserviertDBInstance)
- Neustart DBInstance (DBInstance)
- RemoveSourceIdentifierFromSubscription (EventSubscription)
- Wiederherstellen DBInstance von DBSnapshot (DBInstance)
- Wiederherstellen DBInstance ToPointInTime (DBInstance)
- Widerrufen DBSecurity GroupIngress (DBSecurityGruppe)
- Amazon Redshift
 - AuthorizeClusterSecurityGroupIngress (ClusterSecurityGroup)
 - AuthorizeSnapshotAccess (Schnappschuss)
 - CopyClusterSnapshot (Schnappschuss)
 - CreateCluster (Cluster)
 - CreateClusterParameterGroup (ClusterParameterGroup)
 - CreateClusterSecurityGroup (ClusterSecurityGroup)
 - CreateClusterSnapshot (Schnappschuss)
 - CreateClusterSubnetGroup (ClusterSubnetGroup)
 - CreateEventSubscription (EventSubscription)
 - CreateHsmClientCertificate (HsmClientCertificate)
 - CreateHsmConfiguration (HsmConfiguration)
 - DeleteCluster (Cluster)
 - DeleteClusterSnapshot (Schnappschuss)
 - DescribeDefaultClusterParameters (DefaultClusterParameters)
 - DisableSnapshotCopy (Cluster)
 - EnableSnapshotCopy (Cluster)
 - **ModifyCluster (Cluster)**
- ModifyClusterSubnetGroup (ClusterSubnetGroup)

- `ModifyEventSubscription` (`EventSubscription`)
- `ModifySnapshotCopyRetentionPeriod` (`Cluster`)
- `PurchaseReservedNodeOffering` (`ReservedNode`)
- `RebootCluster` (`Cluster`)
- `RestoreFromClusterSnapshot` (`Cluster`)
- `RevokeClusterSecurityGroupIngress` (`ClusterSecurityGroup`)
- `RevokeSnapshotAccess` (Schnappschuss)
- `RotateEncryptionKey` (`Cluster`)

Enum-Klassen wurden entfernt

Wir haben die Enum-Klassen entfernt (z. B. `Aws\S3\Enum\CannedAc1`), die es in Version 2 des AWS SDK für PHP gab. Aufzählungen waren konkrete Klassen innerhalb der öffentlichen API des SDK, die Konstanten enthielten, die Gruppen von gültigen Parameterwerten darstellten. Da diese Aufzählungen spezifisch für API-Versionen sind, sich im Laufe der Zeit ändern können, mit PHP reservierten Wörtern kollidieren können und letztendlich nicht sehr nützlich sind, haben wir sie in Version 3 entfernt. Dies unterstützt die datengesteuerte und von der API-Version unabhängige Natur von Version 3.

Verwenden Sie anstelle von Werten aus Enum-Objekten direkt die literalen Werte (z. B. `CannedAc1::PUBLIC_READ` → `'public-read'`).

Feinkörnige Ausnahmeklassen wurden entfernt

Wir haben die differenzierten Ausnahmeklassen entfernt, die es in den Namespaces jedes Services gab (zum Beispiel `Aws\Rds\Exception\{SpecificError}Exception`). Die Gründe dafür sind sehr ähnlich denjenigen, aus denen wir Enums entfernt haben. Die Ausnahmen, die von einem Service oder einer Operation aufgeworfen werden, sind abhängig davon, welche API-Version verwendet wird (sie können von Version zu Version wechseln). Außerdem ist die vollständige Liste der Ausnahmen, die durch eine bestimmte Operation ausgegeben werden können, nicht verfügbar, wodurch die differenzierten Ausnahmeklassen der Version 2 unvollständig wurden.

Verarbeiten Sie Fehler, indem Sie die Root-Ausnahmeklasse für jeden Service abfangen (z. B. `Aws\Rds\Exception\RdsException`). Sie können die `getAwsErrorCode()`-Methode der Ausnahme verwenden, um auf bestimmte Fehlercodes zu prüfen. Dies ist funktional äquivalent zum Abfangen verschiedener Ausnahmeklassen, bietet aber diese Funktion, ohne das SDK aufzublähen.

Statische Fassadenklassen wurden entfernt

In Version 2 von gab es eine obskure AWS SDK für PHP, von Laravel inspirierte Funktion, mit der Sie die `Aws` Klasse aufrufen konnten, um den statischen Zugriff `enableFacades()` auf die verschiedenen Service-Clients zu ermöglichen. Diese Funktion widerspricht den Best Practices von PHP, und wir haben vor über einem Jahr aufgehört, sie zu dokumentieren. In Version 3 wurde diese Funktion vollständig entfernt. Rufen Sie Ihre Client-Objekte aus dem `Aws\Sdk`-Objekt ab und verwenden Sie sie als Objekt-Instances, nicht als statische Klassen.

Paginatoren ersetzen Iteratoren

Version 2 von AWS SDK für PHP hatte eine Funktion namens ** Iteratoren**. Dies waren Objekte, die zur Iteration paginierter Ergebnisse verwendet wurden. Eine Beschwerde, die wir dazu hatten, war, dass sie nicht flexibel genug waren, da der Iterator nur bestimmte Werte von jedem Ergebnis ausgab. Wenn es andere Werte gab, die Sie aus den Ergebnissen brauchten, konnten Sie diese nur über Ereignis-Listener abrufen.

In Version 3 wurden Iteratoren durch [Paginatoren](#) ersetzt. Ihr Zweck ist ähnlich, aber Paginatoren sind viel flexibler. Dies liegt daran, dass sie Ergebnisobjekte anstelle von Werten aus einer Antwort liefern.

Die folgenden Beispiele zeigen, wie sich Paginatoren von Iteratoren unterscheiden, indem demonstriert wird, wie paginierte Ergebnisse für die `S3 ListObjects`-Operation in Version 2 und Version 3 abgerufen werden.

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

Paginator-Objekte verfügen über eine `search()` Methode, mit der Sie [JMESPath](#)-Ausdrücke verwenden können, um Daten einfacher aus der Ergebnismenge zu extrahieren.

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

Note

Die `getIterator()`-Methode wird weiterhin unterstützt, damit ein reibungsloser Übergang zu Version 3 möglich ist, aber wir möchten Sie bitten, ab jetzt Paginatoren in Ihrem Code zu verwenden.

Viele Abstraktionen auf höherer Ebene haben sich geändert

Generell wurden viele der übergeordneten Abstraktionen (neben den Clients auch Service-spezifische Helferobjekte) verbessert oder aktualisiert. Einige wurden sogar entfernt.

- Aktualisiert:
 - Die Art und Weise, wie Sie [mehrteilige Uploads in Amazon S3](#) verwenden, hat sich geändert. Amazon Glacier Multipart Upload wurde auf ähnliche Weise geändert.
 - Die Art und Weise, [Amazon S3 vorsigniert](#) zu erstellen, URLs hat sich geändert.
 - Der `Aws\S3\Sync`-Namespace wurde durch `Aws\S3\Transfer` ersetzt. Die Methoden `S3Client::uploadDirectory()` und `S3Client::downloadBucket()` sind noch verfügbar, verwenden aber andere Optionen. Weitere Informationen finden Sie in der Dokumentation für [Amazon S3 Transfer Manager mit AWS SDK für PHP Version 3](#).
 - `Aws\S3\Model\ClearBucket` und `Aws\S3\Model\DeleteObjectsBatch` wurden durch `Aws\S3\BatchDelete` und `S3Client::deleteMatchingObjects()` ersetzt.
 - Die Optionen und das Verhalten für die [Verwendung des DynamoDB-Sessionshandlers mit AWS SDK für PHP Version 3](#) haben sich geringfügig geändert.
 - Der `Aws\DynamoDb\Model\BatchRequest`-Namespace wurde durch `Aws\DynamoDb\WriteRequestBatch` ersetzt. Weitere Informationen finden Sie in der Dokumentation für [DynamoDB WriteRequestBatch](#).

- Der `Aws\Ses\SesClient` verarbeitet jetzt die base64-Verschlüsselung der `RawMessage`, wenn die Operation `SendRawEmail` verwendet wird.
- Entfernt:
 - [Amazon DynamoDB `ItemAttribute`, und `ItemIterator` Klassen](#) — Diese waren zuvor [in Version 2.7.0 veraltet](#).
 - Amazon SNS SNS-Nachrichtenvalidator — Dies ist jetzt [ein separates, schlankes Projekt](#), für das das SDK nicht als Abhängigkeit erforderlich ist. Dieses Projekt ist jedoch in der Phar- und ZIP-Distribution des SDK enthalten. Eine Anleitung für die ersten Schritte finden Sie [im AWS PHP-Entwicklungs-Blog](#).
 - Amazon S3 `AcpBuilder` und zugehörige Objekte wurden entfernt.

Vergleich von Codebeispielen aus beiden Versionen des SDK

Die folgenden Beispiele zeigen, wie sich die Verwendung von Version 3 von von Version 2 AWS SDK für PHP unterscheidet.

Beispiel: Amazon S3 **ListObjects** S3-Betrieb

Ab Version 2 des 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";
    }
}
```

Ab Version 3 des SDK

Wichtigste Unterschiede:

- Verwendung von `new` anstelle von `factory()` zur Instanziierung des Clients.
- Die Optionen `'version'` und `'region'` sind bei der Instanziierung erforderlich.

```
<?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";
}
```

Beispiel: Instanzieren eines Clients mit globaler Konfiguration

Ab Version 2 des 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.
```

Ab Version 3 des SDK

Wichtigste Unterschiede:

- Verwendung der `Aws\Sdk`-Klasse statt `Aws\Common\Aws`.
- Es gibt keine Konfigurationsdatei. Verwenden Sie stattdessen ein Array für die Konfiguration.

- Die Option 'version' ist bei der Instanziierung erforderlich.
- Verwendung der create<Service>()-Methoden statt 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.
```

Sicherheit für AWS SDK für PHP

Cloud-Sicherheit genießt bei Amazon Web Services (AWS) höchste Priorität. Als AWS -Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat. Sicherheit ist eine gemeinsame Verantwortung zwischen Ihnen AWS und Ihnen. Im [Modell der übergreifenden Verantwortlichkeit](#) wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der alle in der AWS Cloud angebotenen Dienste ausgeführt werden, und für die Bereitstellung von Diensten, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei uns höchste Priorität AWS, und die Wirksamkeit unserer Sicherheit wird im Rahmen der [AWS Compliance-Programme](#) regelmäßig von externen Prüfern getestet und verifiziert.

Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem von Ihnen genutzten AWS Dienst und anderen Faktoren, wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Themen

- [Datenschutz in AWS SDK für PHP Version 3](#)
- [Identitäts- und Zugriffsverwaltung](#)
- [Überprüfung der Einhaltung der Vorschriften für dieses AWS Produkt oder diese Dienstleistung](#)
- [Ausfallsicherheit für dieses AWS Produkt oder diese Dienstleistung](#)
- [Sicherheit der Infrastruktur für dieses AWS Produkt oder diesen Service](#)
- [Amazon S3 Encryption Client Migration \(V1 zu V2\) in der AWS SDK für PHP Version 3](#)
- [Amazon S3 Encryption Client Migration \(V2 zu V3\) in der AWS SDK für PHP Version 3](#)

Datenschutz in AWS SDK für PHP Version 3

Das AWS [Modell](#) der gilt für den Datenschutz in. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden

Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag [AWS -Modell der geteilten Verantwortung und in der DSGVO](#) im AWS - Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit AWS SDK für PHP Version 3 oder einer anderen Version arbeiten und dabei die Konsole, die API oder AWS-Services verwenden. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service, den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS IAM ist ein Programm AWS-Service, das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [Wie AWS-Services arbeiten Sie mit IAM](#)
- [Fehlerbehebung bei AWS Identität und Zugriff](#)

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. AWS

Dienstbenutzer — Wenn Sie dies AWS-Services für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr AWS Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Falls Sie auf eine Funktion in der von Ihnen verwendeten Version nicht zugreifen können AWS, finden [Fehlerbehebung bei AWS Identität und Zugriff](#) Sie weitere Informationen in der Bedienungsanleitung des AWS-Service Geräts, das Sie verwenden.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM verwenden kann AWS, finden Sie in der Benutzeranleitung des von AWS-Service Ihnen verwendeten.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der AWS-Service von Ihnen verwendeten.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter [So melden Sie sich bei Ihrem AWS-Konto an](#) im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter [AWS Signature Version 4 for API requests](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Tasks that require root user credentials](#) im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können](#).

Eine [IAM-Gruppe](#) spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#) AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter [Methoden, um eine Rolle zu übernehmen](#) im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für Verbundbenutzerzugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, dienstübergreifenden Zugriff und Anwendungen, die auf Amazon ausgeführt werden. EC2 Weitere Informationen finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an Identitäten oder Ressourcen anhängen. AWS Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcen und unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter [Choose between managed policies and inline policies](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#).

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter [Übersicht über ACLs die Zugriffskontrollliste \(ACL\)](#) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
- **Richtlinien zur Dienstkontrolle (SCPs)** — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter [Service-Kontrollrichtlinien](#) im AWS Organizations -Benutzerhandbuch.
- **Richtlinien zur Ressourcenkontrolle (RCPs)** — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter [Richtlinien zur Ressourcenkontrolle \(RCPs\)](#).
- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter [Bewertungslogik für Richtlinien](#) im IAM-Benutzerhandbuch.

Wie AWS-Services arbeiten Sie mit IAM

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im [AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren](#).

Informationen zur Verwendung bestimmter Dienste AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des jeweiligen Dienstes.

Fehlerbehebung bei AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS und IAM auftreten können.

Themen

- [Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS](#)
- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen](#)

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *my-example-widget*-Ressource anzuzeigen, jedoch nicht über `aws:GetWidget`-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `aws:GetWidget`-Aktion auf die *my-example-widget*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWS übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen AWS unterstützt werden, finden Sie unter [Wie AWS-Services arbeiten Sie mit IAM](#)
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto, den Sie besitzen](#).
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Überprüfung der Einhaltung der Vorschriften für dieses AWS Produkt oder diese Dienstleistung

Informationen darüber, ob AWS-Service ein [AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter](#) . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#) .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der [AWS Sicherheitsdokumentation](#).

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

Ausfallsicherheit für dieses AWS Produkt oder diese Dienstleistung

Die AWS globale Infrastruktur basiert auf AWS-Regionen Availability Zones.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind.

Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

Sicherheit der Infrastruktur für dieses AWS Produkt oder diesen Service

Dieses AWS Produkt oder dieser Dienst verwendet Managed Services und ist daher durch die AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf dieses AWS Produkt oder diesen Service zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS -Security-Token-Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Dieses AWS Produkt oder dieser Service folgt dem [Modell der gemeinsamen Verantwortung](#) in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der [Seite mit der Dokumentation zur AWS Servicesicherheit](#) und den [AWS Services, für die das AWS Compliance-Programm zur Einhaltung der](#) Vorschriften zuständig ist.

Amazon S3 Encryption Client Migration (V1 zu V2) in der AWS SDK für PHP Version 3

Note

Wenn Sie Version 2 (V2) des Amazon S3 S3-Verschlüsselungsclients verwenden und zu Version 3 (V3) migrieren möchten, finden Sie weitere Informationen unter [Amazon S3 Encryption Client Migration \(V2 zu V3\) in der AWS SDK für PHP Version 3](#).

In diesem Thema erfahren Sie, wie Sie Ihre Anwendungen von Version 1 (V1) des Amazon Simple Storage Service (Amazon S3) -Verschlüsselungsclients auf Version 2 (V2) migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen.

Überblick über die Migration

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Stellen Sie zunächst eine aktualisierte Version von AWS SDK für PHP für Ihre Anwendung bereit. Dadurch können bestehende V1-Verschlüsselungsclients Objekte entschlüsseln, die von den neuen V2-Clients geschrieben wurden. Wenn Ihre Anwendung mehrere verwendet AWS SDKs, müssen Sie jedes SDK separat aktualisieren.
2. Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V2. Sobald alle Ihre V1-Verschlüsselungsclients neue Formate lesen können, können Sie Ihre vorhandenen Verschlüsselungs- und Entschlüsselungsclients auf ihre jeweiligen V2-Versionen migrieren.

Aktualisieren Sie bestehende Clients, um neue Formate lesen zu können

Der V2-Verschlüsselungsclient verwendet Verschlüsselungsalgorithmen, die ältere Versionen des Clients nicht unterstützen. Der erste Schritt der Migration besteht darin, Ihre V1-Entschlüsselungsclients auf die neueste SDK-Version zu aktualisieren. Nach Abschluss dieses Schritts können die V1-Clients Ihrer Anwendung Objekte entschlüsseln, die mit V2-Verschlüsselungsclients verschlüsselt wurden. Nachfolgend finden Sie Einzelheiten zu den einzelnen Hauptversionen von AWS SDK für PHP

Aktualisierung von AWS SDK für PHP Version 3

Version 3 ist die neueste Version von AWS SDK für PHP. Um diese Migration abzuschließen, müssen Sie Version 3.148.0 oder höher des `aws/aws-sdk-php` Pakets verwenden.

Installation über die Befehlszeile

Aktualisieren Sie bei Projekten, die mit Composer installiert wurden, in der Composer-Datei das SDK-Paket auf Version 3.148.0 des SDK und führen Sie dann den folgenden Befehl aus.

```
composer update aws/aws-sdk-php
```

Installation mithilfe der Phar- oder Zip-Datei

Verwenden Sie eine der folgenden Methoden: Stellen Sie sicher, dass Sie die aktualisierte SDK-Datei an dem für Ihren Code erforderlichen Speicherort ablegen, der durch die `require`-Anweisung bestimmt wird.

Laden Sie für Projekte, die mit der Phar-Datei installiert wurden, die aktualisierte Datei herunter:

[aws.phar](#).

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

Laden Sie für Projekte, die mit der Zip-Datei installiert wurden, die aktualisierte Datei herunter: .

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

Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients auf V2

Nachdem Sie Ihre Clients so aktualisiert haben, dass sie die neuen Verschlüsselungsformate lesen können, können Sie Ihre Anwendungen auf die V2-Verschlüsselungs- und Entschlüsselungsclients aktualisieren. Die folgenden Schritte zeigen Ihnen, wie Sie Ihren Code erfolgreich von V1 auf V2 migrieren können.

Anforderungen für die Aktualisierung auf V2-Clients

1. Der AWS KMS Verschlüsselungskontext muss an die `S3EncryptionClientV2::putObjectAsync` Methoden `S3EncryptionClientV2::putObject` und übergeben werden. AWS KMS Der Verschlüsselungskontext ist ein assoziatives Array von Schlüssel-Wert-Paaren, die Sie dem Verschlüsselungskontext für AWS KMS die Schlüsselverschlüsselung hinzufügen müssen. Wenn kein zusätzlicher Kontext erforderlich ist, können Sie ein leeres Array übergeben.
2. `@SecurityProfile` muss an die `getObject` und `getObjectAsync` Methoden in übergeben werden `S3EncryptionClientV2`. `@SecurityProfile` ist ein neuer obligatorischer Parameter der `getObject...` Methoden. Wenn auf gesetzt 'V2', können nur Objekte entschlüsselt werden, die im V2-kompatiblen Format verschlüsselt sind. Wenn dieser Parameter auf gesetzt wird, können 'V2_AND_LEGACY' auch Objekte entschlüsselt werden, die im V1-kompatiblen Format verschlüsselt wurden. Um die Migration zu unterstützen, setzen Sie ihn auf. `@SecurityProfile` 'V2_AND_LEGACY' 'V2' Nur für die Entwicklung neuer Anwendungen verwenden.
3. (optional) Fügen Sie den `@KmsAllowDecryptWithAnyCmk` Parameter in das `S3EncryptionClientV2::getObjectAsync*` methods. ein `S3EncryptionClientV2::getObject` und Ein neuer Parameter wurde hinzugefügt, der aufgerufen wurde `@KmsAllowDecryptWithAnyCmk`. Wenn Sie diesen Parameter so einstellen, dass die Entschlüsselung ohne Angabe eines KMS-Schlüssels `true` aktiviert wird. Der Standardwert ist `false`.
4. Wenn bei der Entschlüsselung mit einem V2-Client der `@KmsAllowDecryptWithAnyCmk` Parameter `true` für die "`getObject...`" Methodenaufrufen nicht auf festgelegt ist, `kms-key-id` muss an den `KmsMaterialsProviderV2` Konstruktor übergeben werden.

Beispiele für Migrationen

Beispiel 1: Migration zu V2-Clients

Vor der Migration

```
use Aws\S3\Crypto\S3EncryptionClient;
use Aws\S3\S3Client;

$encryptionClient = new S3EncryptionClient(
    new S3Client([
```

```
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => 'latest',  
    ])  
);
```

Nach der Migration

```
use Aws\S3\Crypto\S3EncryptionClientV2;  
use Aws\S3\S3Client;  
  
$encryptionClient = new S3EncryptionClientV2(  
    new S3Client([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => 'latest',  
    ])  
);
```

Beispiel 2: Verwendung mit AWS KMS kms-key-id

Note

In diesen Beispielen werden Importe und Variablen verwendet, die in Beispiel 1 definiert sind. Beispiel, `$encryptionClient`.

Vor der Migration

```
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,
]);
```

Nach der Migration

```
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,
]);
```

Amazon S3 Encryption Client Migration (V2 zu V3) in der AWS SDK für PHP Version 3

Note

Wenn Sie Version 1 (V1) des Amazon S3 S3-Verschlüsselungsclients verwenden, müssen Sie zuerst auf Version 2 (V2) migrieren, bevor Sie auf Version 3 (V3) migrieren. Siehe [Amazon S3 Encryption Client Migration \(V1 zu V2\) in der AWS SDK für PHP Version 3](#).

In diesem Thema erfahren Sie, wie Sie Ihre Anwendungen von Version 2 (V2) des Amazon Simple Storage Service (Amazon S3) -Verschlüsselungsclients auf Version 3 (V3) migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen. Version 3 führt AES GCM mit Key Commitment- und Commitment-Richtlinien ein, um die Sicherheit zu erhöhen und vor der Manipulation von Datenschlüsseln zu schützen.

Überblick über die Migration

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Stellen Sie zunächst eine aktualisierte Version von AWS SDK für PHP für Ihre Anwendung bereit. Dadurch können bestehende V2-Verschlüsselungsclients Objekte entschlüsseln, die von den neuen V3-Clients geschrieben wurden. Wenn Ihre Anwendung mehrere verwendet AWS SDKs, müssen Sie jedes SDK separat aktualisieren.

2. Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V3. Sobald alle Ihre V2-Verschlüsselungsclients neue Formate lesen können, können Sie Ihre vorhandenen Verschlüsselungs- und Entschlüsselungsclients auf ihre jeweiligen V3-Versionen migrieren.

V3-Konzepte verstehen

Version 3 des Amazon S3 S3-Verschlüsselungsclients führt zwei wichtige Sicherheitsverbesserungen ein: Commitment Policies und den Algorithmus AES GCM with Key Commitment. Das Verständnis dieser Konzepte ist für eine erfolgreiche Migration unerlässlich.

Politik des Engagements

Eine Commitment-Richtlinie steuert, wie der Verschlüsselungsclient bei Verschlüsselungs- und Entschlüsselungsvorgängen mit der Schlüsselzusage umgeht. Version 3 bietet drei Richtlinienoptionen:

FORBID_ENCRYPT_ALLOW_DECRYPT

Verschlüsselungsverhalten: Verschlüsselt Objekte ohne Schlüsselbindung.

Entschlüsselungsverhalten: Ermöglicht die Entschlüsselung von Objekten, die mit oder ohne Schlüsselbindung verschlüsselt wurden.

Auswirkungen auf die Sicherheit: Diese Richtlinie erzwingt keine Schlüsselzuweisung für neu verschlüsselte Objekte, wodurch Datenschlüssel manipuliert werden können. Verwenden Sie diese Richtlinie nur in der ersten Migrationsphase, wenn Sie die Kompatibilität mit V2-Clients aufrechterhalten müssen.

Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können von allen V2- und V3-Implementierungen gelesen werden.

REQUIRE_ENCRYPT_ALLOW_DECRYPT

Verschlüsselungsverhalten: Verschlüsselt Objekte mit Schlüsselzusage mithilfe des ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY Algorithmus.

Entschlüsselungsverhalten: Ermöglicht die Entschlüsselung von Objekten, die mit oder ohne Schlüsselbindung verschlüsselt wurden.

Auswirkungen auf die Sicherheit: Diese Richtlinie bietet verbesserte Sicherheit für neu verschlüsselte Objekte und behält gleichzeitig die Möglichkeit, vorhandene Objekte zu lesen. Dies ist die empfohlene Richtlinie für die meisten Migrationsszenarien.

Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können nur von V3 und den neuesten V2-Implementierungen gelesen werden.

Überlegungen zur Migration: Stellen Sie vor der Verwendung dieser Richtlinie sicher, dass alle Clients, die die verschlüsselten Objekte lesen müssen, auf Version 3 oder die neueste Version V2 aktualisiert wurden.

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

Verschlüsselungsverhalten: Verschlüsselt Objekte mit Schlüsselzuweisung mithilfe des `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus.

Entschlüsselungsverhalten: Erlaubt nur die Entschlüsselung von Objekten, die mit Schlüsselzuweisung verschlüsselt wurden. Objekte, die ohne Schlüsselzuweisung verschlüsselt wurden, können nicht entschlüsselt werden.

Auswirkungen auf die Sicherheit: Diese Richtlinie bietet ein Höchstmaß an Sicherheit, indem sie sowohl bei der Verschlüsselung als auch bei der Entschlüsselung eine Schlüsselbindung erzwingt. Verwenden Sie diese Richtlinie erst, nachdem alle Objekte zur Verwendung von Key Commitment migriert wurden.

Versionskompatibilität: Nur V3-Implementierungen können diese Richtlinie verwenden. Der Versuch, V1- oder V2-verschlüsselte Objekte mit dieser Richtlinie zu entschlüsseln, schlägt fehl.

Überlegungen zur Migration: Diese Richtlinie sollte erst verwendet werden, nachdem die vollständige Migration abgeschlossen und alle vorhandenen Objekte mit Schlüsselzusage erneut verschlüsselt wurden.

AES GCM mit Key Commitment

Der Algorithmus AES GCM with Key Commitment

(`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) ist ein neuer Verschlüsselungsalgorithmus, der in Version 3 eingeführt wurde und Schutz vor Angriffen zur Manipulation von Datenschlüsseln bietet.

Verbesserung der Sicherheit: `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Schützt vor Manipulation von Datenschlüsseln, indem der Datenschlüssel kryptografisch an den verschlüsselten Inhalt gebunden wird. Dadurch wird verhindert, dass Angreifer bei der Entschlüsselung einen anderen Datenschlüssel einsetzen, was zur unbeabsichtigten Entschlüsselung von Daten führen könnte.

Versionskompatibilität: Mit verschlüsselte Objekte

`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` können nur mit V3 und den neuesten V2-Implementierungen des Amazon S3 S3-Verschlüsselungsclients entschlüsselt werden. V1-Clients können mit diesem Algorithmus verschlüsselte Objekte nicht entschlüsseln.

Important

Upgrade-Anforderung: Bevor Sie die Verschlüsselung mit `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (mithilfe der Richtlinien `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `REQUIRE_ENCRYPT_REQUIRE_DECRYPT`) aktivieren, müssen Sie sicherstellen, dass alle Clients, die die verschlüsselten Objekte lesen müssen, auf Version 3 aktualisiert wurden. Wenn nicht alle Lesegeräte aktualisiert werden, schlägt die Entschlüsselung von Objekten fehl, die mit Key Commitment verschlüsselt wurden.

Aktualisieren Sie vorhandene Clients, damit sie neue Formate lesen können

Der V3-Verschlüsselungsclient verwendet Verschlüsselungsalgorithmen und wichtige Funktionen, die von älteren Versionen des Clients nicht unterstützt werden. Der erste Schritt der Migration besteht darin, Ihre V2-Entschlüsselungsclients auf die neueste SDK-Version zu aktualisieren. Nach Abschluss dieses Schritts können die V2-Clients Ihrer Anwendung Objekte entschlüsseln, die mit V3-Verschlüsselungsclients verschlüsselt wurden. Nachfolgend finden Sie Einzelheiten zu den einzelnen Installationsmethoden von AWS SDK für PHP

Die neueste SDK-Version erstellen und installieren

Um diese Migration abzuschließen, müssen Sie die neueste Version des `aws/aws-sdk-php` Pakets verwenden, das Unterstützung für V3-Verschlüsselungsclients enthält.

Installation über Composer

Aktualisieren Sie bei Projekten, die mit Composer installiert wurden, in der Composer-Datei das SDK-Paket auf die neueste Version des SDK und führen Sie dann den folgenden Befehl aus.

```
composer update aws/aws-sdk-php
```

Installation mithilfe der Phar- oder Zip-Datei

Verwenden Sie eine der folgenden Methoden: Stellen Sie sicher, dass Sie die aktualisierte SDK-Datei an dem für Ihren Code erforderlichen Speicherort ablegen, der durch die require-Anweisung bestimmt wird.

Laden Sie für Projekte, die mit der Phar-Datei installiert wurden, die aktualisierte Datei herunter: [aws.phar](#).

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

Laden Sie für Projekte, die mit der Zip-Datei installiert wurden, die aktualisierte Datei herunter: .

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

Anwendungen erstellen, installieren und bereitstellen

Nachdem Sie das SDK aktualisiert haben, erstellen Sie Ihre Anwendung neu und stellen Sie sie erneut bereit, um sicherzustellen, dass alle Komponenten die aktualisierte Version verwenden. Dieser Schritt ist wichtig, um sicherzustellen, dass Ihre V2-Clients Objekte lesen können, die von V3-Clients verschlüsselt wurden.

Folgen Sie den Standardbereitstellungsverfahren Ihres Unternehmens, um die aktualisierte Anwendung einzuführen. Stellen Sie sicher, dass alle Instanzen Ihrer Anwendung aktualisiert sind, bevor Sie mit der Migration Ihrer Verschlüsselungs- und Entschlüsselungsclients auf Version 3 fortfahren.

Stellen Sie nach der Bereitstellung sicher, dass Ihre Anwendung weiterhin vorhandene Objekte entschlüsseln kann und dass während des normalen Betriebs keine Fehler auftreten. Dadurch wird bestätigt, dass das SDK-Update erfolgreich war und Ihre Anwendung für die nächste Migrationsphase bereit ist.

Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf Version 3

Nachdem Sie Ihre Clients so aktualisiert haben, dass sie die neuen Verschlüsselungsformate lesen können, können Sie Ihre Anwendungen auf die V3-Verschlüsselungs- und Entschlüsselungsclients aktualisieren. Die folgenden Beispiele zeigen Ihnen, wie Sie Ihren Code erfolgreich von V2 auf V3 migrieren können.

Verwenden von V3-Verschlüsselungsclients

V3 führt die `S3EncryptionClientV3` Klasse ein und ersetzt `KmsMaterialsProviderV3` die V2-Äquivalente. Die wichtigsten Unterschiede in V3 sind:

- V3 verwendet `KmsMaterialsProviderV3` (wie V2), überprüft aber den Verschlüsselungskontext beim Entschlüsseln von Objekten in `GetObject` Aufrufen.
- V3 führt Commitment-Richtlinien ein, um das Ver- und Entschlüsselungsverhalten zu kontrollieren.

Beispiel: Migration von V2 zu V3 mit KMS-Verschlüsselung

Vor der Migration (V2)

```
use Aws\S3\Crypto\S3EncryptionClientV2;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV2;
use Aws\Kms\KmsClient;

$encryptionClient = new S3EncryptionClientV2(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV2(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
```

```

);

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

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

$result = $encryptionClient->getObject([
    '@KmsAllowDecryptWithAnyCmk' => true,
    '@SecurityProfile' => 'V2_AND_LEGACY',
    '@CommitmentPolicy' => 'FORBID_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);

```

Während der Migration (V3 mit Abwärtskompatibilität)

```

use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

```



```
// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

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

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

$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRED_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Nach der Migration (V3 mit zentraler Verpflichtung)

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
```

```
use Aws\Kms\KmsClient;

// Create V3 encryption client
$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

// Create encryption materials
$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

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

$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    // Use the commitment policy (REQUIRED_ENCRYPT_REQUIRED_DECRYPT)
    // This encrypts with key commitment and does not decrypt V2 objects
    '@CommitmentPolicy' => 'REQUIRED_ENCRYPT_REQUIRED_DECRYPT',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3',
    // Use the commitment policy (REQUIRED_ENCRYPT_REQUIRED_DECRYPT)
```

```
// This encrypts with key commitment and does not decrypt V2 objects
'@CommitmentPolicy' => 'REQUIRED_ENCRYPT_REQUIRED_DECRYPT',
'@MaterialsProvider' => $materialsProvider,
'@CipherOptions' => $cipherOptions,
'Bucket' => $bucket,
'Key' => $key,
]);
```

Hauptunterschiede in V3:

- Verwenden Sie `KmsMaterialsProviderV3` anstelle von `KmsMaterialsProviderV2`.
- Der `@KmsEncryptionContext` Parameter ist weiterhin für `putObject` Operationen erforderlich
- Der `@KmsEncryptionContext` Parameter ist für `getObject` Operationen optional und überprüft, ob der angegebene Verschlüsselungskontext mit dem im Objekt übereinstimmt.
- Der `@SecurityProfile` Parameter steuert, welche Verschlüsselungsversionen entschlüsselt werden können. Wird auf gesetzt, `'V3_AND_LEGACY'` um das Lesen von V1- und V2-verschlüsselten Objekten während der Migration zu unterstützen
- Der `@CommitmentPolicy` Parameter steuert die Commitment-Richtlinie für diesen Vorgang. Wird auf gesetzt `'FORBID_ENCRYPT_ALLOW_DECRYPT'`, um das Lesen verschlüsselter Objekte ohne Verpflichtung während der Migration zu unterstützen

Weitere Beispiele

Die folgenden Beispiele zeigen zusätzliche Konfigurationsoptionen, die in Version 3 verfügbar sind und Ihnen helfen können, den Migrationsprozess zu verwalten und das Verschlüsselungsverhalten zu kontrollieren.

Aktivieren des Legacy-System--Supports

Während der Migration müssen Sie möglicherweise Objekte entschlüsseln, die mit Version 1 oder V2 des Amazon S3 S3-Verschlüsselungsclients verschlüsselt wurden. Der `@SecurityProfile` Parameter steuert, welche Verschlüsselungsversionen Ihr V3-Client entschlüsseln kann.

Wann sollte diese Konfiguration verwendet werden? Verwenden Sie das `'V3_AND_LEGACY'` Sicherheitsprofil, wenn Ihre Anwendung Objekte lesen muss, die von V1- oder V2-Clients verschlüsselt wurden. Dies ist während der Migrationsphase üblich, wenn Sie eine Mischung aus alten und neuen verschlüsselten Objekten in Ihren Buckets haben.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

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

// Decrypt objects encrypted with V1, V2, or V3
$result = $encryptionClient->getObject([
    '@SecurityProfile' => 'V3_AND_LEGACY',
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_ALLOW_DECRYPT',
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    'Bucket' => $bucket,
    'Key' => $key,
]);
```

Der Parameter `@SecurityProfile` akzeptiert die folgenden Werte:

- 'V3' (Standard): Entschlüsseln Sie nur Objekte, die mit V3 mithilfe von Key Commitment verschlüsselt wurden
- 'V3_AND_LEGACY': Entschlüsselt Objekte, die mit V1, V2 oder V3 verschlüsselt wurden

Important

Nachdem Sie Ihre Migration abgeschlossen und alle Objekte mit V3 erneut verschlüsselt haben, sollten Sie den `@SecurityProfile` Parameter entfernen oder auf `default` einstellen, um maximale Sicherheit 'V3' zu gewährleisten.

Speichermethode konfigurieren

Der Amazon S3 S3-Verschlüsselungscient kann Verschlüsselungsmetadaten auf zwei Arten speichern: in den Metadaten-Headern des Objekts oder in einer separaten Anweisungsdatei. Der `@MetadataStrategy` Parameter steuert, welche Speichermethode verwendet wird.

Wann Sie diese Konfiguration verwenden sollten: Verwenden Sie diese Konfiguration, 'INSTRUCTION_FILE' wenn Sie die ursprünglichen Objektmetadaten beibehalten müssen oder wenn Sie mit Objekten arbeiten, für die Metadaten Größenbeschränkungen gelten. Verwenden Sie 'METADATA' (Standardeinstellung) für einfachere Bereitstellungen, bei denen Verschlüsselungsmetadaten zusammen mit dem Objekt gespeichert werden können.

```
use Aws\S3\Crypto\S3EncryptionClientV3;
use Aws\S3\S3Client;
use Aws\Crypto\KmsMaterialsProviderV3;
use Aws\Kms\KmsClient;

$kmsKeyId = 'kms-key-id';
$materialsProvider = new KmsMaterialsProviderV3(
    new KmsClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ]),
    $kmsKeyId
);

$encryptionClient = new S3EncryptionClientV3(
    new S3Client([
```

```
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => 'latest',
    ])
);

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

// Store encryption metadata in a separate instruction file
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'INSTRUCTION_FILE',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);

// Store encryption metadata in object headers (default)
$encryptionClient->putObject([
    '@MaterialsProvider' => $materialsProvider,
    '@CipherOptions' => $cipherOptions,
    '@CommitmentPolicy' => 'REQUIRE_ENCRYPT_REQUIRE_DECRYPT',
    '@MetadataStrategy' => 'METADATA',
    '@KmsEncryptionContext' => ['context-key' => 'context-value'],
    'Bucket' => $bucket,
    'Key' => $key,
    'Body' => fopen('file-to-encrypt.txt', 'r'),
]);
```

Der Parameter `@MetadataStrategy` akzeptiert die folgenden Werte:

- `'METADATA'` (Standard): Speichert Verschlüsselungsmetadaten in den Metadaten-Headern des Objekts
- `'INSTRUCTION_FILE'`: Speichert Verschlüsselungsmetadaten in einer separaten Befehlsdatei mit dem Suffix `.instruction`

 Note

Bei der Verwendung 'INSTRUCTION_FILE' bietet der Algorithmus AES GCM mit Key Commitment zusätzlichen Schutz vor Manipulation von Datenschlüsseln. Objekte, die 'METADATA' Speicher verwenden, profitieren nicht von diesem zusätzlichen Schutz.

Häufig gestellte Fragen zu AWS SDK für PHP Version 3

Welche Methoden sind auf einem Client verfügbar?

Die AWS SDK für PHP verwendet Dienstbeschreibungen und dynamische [magische `\_\_call\(\)` - Methoden](#), um API-Operationen auszuführen. Eine vollständige Liste der verfügbaren Methoden für einen Web-Service-Client finden Sie in der [API-Dokumentation](#) des Clients.

Was mache ich bei einem cURL SSL-Zertifikatsfehler?

Dieses Problem kann auftreten, wenn Sie ein out-of-date CA-Bundle mit cURL und SSL verwenden. Sie können dieses Problem umgehen, indem Sie das CA-Bundle auf Ihrem Server aktualisieren oder ein weiteres up-to-date CA-Bundle [direkt von der cURL-Website](#) herunterladen.

Standardmäßig AWS SDK für PHP wird das CA-Bundle verwendet, das bei der Kompilierung von PHP konfiguriert wurde. Sie können das von PHP verwendete Standard-CA-Bundle ändern, indem Sie die Konfigurationseinstellung `openssl.cafile` PHP `.ini` auf den Pfad einer CA-Datei auf der Festplatte setzen.

Welche API-Versionen sind für einen Client verfügbar?

Beim Erstellen eines Clients ist eine `version`-Option erforderlich. Eine Liste der verfügbaren API-Versionen finden Sie auf der API-Dokumentationsseite jedes Kunden::aws-php-class:<index.html>. Wenn Sie eine bestimmte API-Version nicht laden können, müssen Sie möglicherweise Ihre Kopie von AWS SDK für PHP aktualisieren.

Sie können die Zeichenkette `latest` für den Konfigurationswert „version“ übergeben, um die aktuellste verfügbare API-Version zu verwenden, die der API-Provider Ihres Clients finden kann (der standardmäßige `api_provider` durchsucht das Verzeichnis `src/data` des SDK nach API-Modellen).

Warning

Wir empfehlen nicht, `latest` in einer Produktionsanwendung zu verwenden, da das Einfügen einer neuen Nebenversion des SDK, die ein API-Update enthält, Ihre Produktionsanwendung beschädigen könnte.

Welche Regionsversionen sind für einen Client verfügbar?

Eine `region`-Option wird beim Erstellen eines Clients benötigt und über einen Zeichenfolgenwert angegeben. Eine Liste der verfügbaren AWS Regionen und Endpunkte finden Sie unter [AWS Regionen und Endpunkte](#) in der Allgemeinen AWS-Referenz

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

Warum kann ich keine Dateien mit Größen über 2 GB hoch- oder herunterladen?

Da der Ganzzahl-Typ von PHP vorzeichensensitiv ist und viele Plattformen 32-Bit-Ganzzahlen verwenden, behandelt AWS SDK für PHP Dateien, die größer als 2 GB sind, auf einem 32-Bit-Stack (wobei der Begriff „Stack“ CPU, Betriebssystem, Webserver und PHP-Binärdatei beinhaltet) nicht korrekt. Dies ist ein [bekanntes Problem von PHP](#). Im Fall von Microsoft Windows 7 unterstützen nur Builds von PHP 7 64-Bit-Ganzzahlwerte.

Die empfohlene Lösung ist die Verwendung eines [64-Bit Linux-Stacks](#), wie die 64-Bit Amazon Linux AMI, mit der neuesten Version von PHP.

Weitere Informationen finden Sie unter [PHP-Dateigröße: Rückgabewerte](#).

Wie kann ich sehen, welche Daten übertragen wurden?

Sie können Debugging-Informationen, einschließlich der über die Leitung gesendeten Daten, mit der Option `debug` in einem Client-Konstruktor erhalten. Wenn diese Option auf `true` gesetzt ist, werden alle Mutationen des ausgeführten Befehls, der gesendeten Anfrage, der empfangenen Antwort und des verarbeiteten Ergebnisses an STDOUT gesendet. Dies umfasst alle Daten, die gesendet und empfangen werden.

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

```
]);
```

Wie kann ich zufällige Header für eine Anforderung festlegen?

Sie können beliebige Header zu einer Service-Operation hinzufügen, indem Sie eine benutzerdefinierte Middleware zur `Aws\HandlerList` eines `Aws\CommandInterface` oder `Aws\ClientInterface` hinzufügen. Das folgende Beispiel zeigt, wie einem bestimmten Amazon S3 `PutObject` S3-Vorgang mithilfe der `Aws\Middleware::mapRequest` Helper-Methode ein `X-Foo-Baz` Header hinzugefügt wird.

Weitere Informationen finden Sie unter [mapRequest](#).

Wie kann ich eine beliebige Anforderung signieren?

Sie können eine beliebige:aws-php-class: PSR-7-Anfrage `<class-psr.http.Message` signieren. `RequestInterface.html>` mithilfe der `SignatureV4`-Klasse des SDK:: aws-php-class `<class-Aws.Signature.SignatureV4.html>`

Ein vollständiges Beispiel dafür finden Sie unter [Signieren von benutzerdefinierten CloudSearch Amazon-Domain-Anfragen mit AWS SDK für PHP Version 3](#).

Wie kann ich einen Befehl vor dem Senden ändern?

Sie können einen Befehl vor dem Senden ändern, indem Sie der `Aws\HandlerList` einer `Aws\CommandInterface` oder `Aws\ClientInterface` eine benutzerdefinierte Middleware hinzufügen. Das folgende Beispiel zeigt, wie benutzerdefinierte Befehlsparameter zu einem Befehl hinzugefügt werden können, bevor er gesendet wird, wobei im Wesentlichen Standardoptionen hinzugefügt werden. Dieses Beispiel verwendet die `Aws\Middleware::mapCommand`-Helfermethode.

Weitere Informationen finden Sie unter [mapCommand](#).

Was ist ein CredentialsException?

Wenn Sie das eine `Aws\Exception\CredentialsException` Weile verwenden, bedeutet das AWS SDK für PHP, dass dem SDK keine Anmeldeinformationen zur Verfügung gestellt wurden und das SDK keine Anmeldeinformationen in der Umgebung finden konnte.

Wenn Sie einen Client ohne Anmeldeinformationen instanziiieren, versucht das SDK bei der ersten Ausführung einer Serviceoperation, Anmeldeinformationen zu finden. Es checkt zuerst einige spezifische Umgebungsvariablen ein und sucht dann nach Anmeldeinformationen für Instance-Profile, die nur auf konfigurierten EC2 Amazon-Instances verfügbar sind. Wenn keine Anmeldeinformationen angegeben oder zu finden sind, wird eine `Aws\Exception\CredentialsException` aufgeworfen.

Wenn dieser Fehler angezeigt wird und Sie beabsichtigen, Anmeldeinformationen für das Instance-Profil zu verwenden, müssen Sie sicherstellen, dass die EC2 Amazon-Instance, auf der das SDK ausgeführt wird, mit einer geeigneten IAM-Rolle konfiguriert ist.

Wenn Sie diesen Fehler sehen und Sie nicht beabsichtigen, Instance-Profil-Anmeldeinformationen zu verwenden, müssen Sie sicher stellen, dass Sie dem SDK korrekt Anmeldeinformationen zur Verfügung stellen.

Weitere Informationen finden Sie unter [Anmeldeinformationen für AWS SDK für PHP Version 3](#).

AWS SDK für PHP Funktioniert das auf HHVM?

Das läuft derzeit AWS SDK für PHP nicht auf HHVM und wird auch nicht in der Lage sein, bis das [Problem mit der Yield-Semantik in HHVM behoben](#) ist.

Wie deaktiviere ich SSL?

Sie können SSL deaktivieren, indem Sie den `scheme`-Parameter in einer Client-Factory-Methode auf „http“ setzen. Beachten Sie, dass nicht alle Services `http`-Zugriff unterstützen. Eine Liste der [AWS Regionen, Endpunkte und](#) der unterstützten Allgemeine AWS-Referenz Schemata finden Sie unter [Regionen und Endpunkte](#) in der.

```
$client = new Aws\DynamoDb\DynamoDbClient([
    'version' => '2012-08-10',
    'region'  => 'us-west-2',
    'scheme'  => 'http'
]);
```

Warning

Da SSL eine Verschlüsselung aller Daten erfordert und mehr TCP-Pakete benötigt, um einen Verbindungs-Handshake durchzuführen als nur TCP, kann die Deaktivierung von

SSL eine kleine Leistungssteigerung bedeuten. Bei deaktiviertem SSL werden jedoch alle Daten unverschlüsselt über die Leitung übertragen. Bevor Sie SSL deaktivieren, müssen Sie die Auswirkungen auf die Sicherheit und die Möglichkeit des Abhörens über das Netzwerk sorgfältig abwägen.

Was tue ich bei einem „Parse-Fehler“?

Die PHP-Engine gibt Parsing-Fehler aus, wenn sie auf Syntax stößt, die sie nicht versteht. Dies ist fast immer der Fall, wenn man versucht, Code auszuführen, der für eine andere Version von PHP geschrieben wurde.

Wenn Sie auf einen Parsing-Fehler stoßen, überprüfen Sie Ihr System und stellen Sie sicher, dass es die [Anforderungen und Empfehlungen des SDK für](#) Version 3 erfüllt. AWS SDK für PHP

Warum dekomprimiert der Amazon S3 S3-Client gezippte Dateien?

Einige HTTP-Handler, einschließlich des standardmäßigen Guzzle 6 HTTP-Handlers, vergrößern standardmäßig komprimierte Antwortrumpfe. Sie können dieses Verhalten überschreiben, indem Sie den [decode_content](#) HTTP-Option auf `false` setzen. Aus Gründen der Abwärtskompatibilität kann diese Voreinstellung nicht geändert werden, wir empfehlen jedoch, die Inhaltsdekodierung auf S3-Client-Ebene zu deaktivieren.

Unter [decode_content](#) finden Sie ein Beispiel dafür, wie Sie automatische Inhaltsdekodierung deaktivieren können.

Wie deaktiviere ich Body-Signing in Amazon S3?

Sie können die Rumpfsignatur deaktivieren, indem Sie den `ContentSHA256`-Parameter im Befehlsobjekt auf `Aws\Signature\S3SignatureV4::UNSIGNED_PAYLOAD` setzen. Dann AWS SDK für PHP wird es als Header `'x-amz-content-sha-256'` und als Body-Prüfsumme in der kanonischen Anfrage verwendet.

```
$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);

```

Wie wird das Wiederholungsschema in AWS SDK für PHP behandelt?

Der AWS SDK für PHP hat eine, die das Wiederholungsverhalten behandelt `RetryMiddleware`. In Bezug auf 5xx HTTP-Statuscodes für Serverfehler versucht das SDK Wiederholungen für 500, 502, 503 und 504.

Ablehnungs-Ausnahmen, wie beispielsweise `RequestLimitExceeded`, `Throttling`, `ProvisionedThroughputExceededException`, `ThrottlingException`, `RequestThrottled` und `BandwidthLimitExceeded`, werden ebenfalls mit Wiederholungen verarbeitet.

Das integriert AWS SDK für PHP auch eine exponentielle Verzögerung mit einem Backoff- und Jitter-Algorithmus in das Wiederholungsschema. Darüber hinaus ist das standardmäßige Wiederholungsverhalten wie 3 für alle Services konfiguriert, mit Ausnahme von Amazon DynamoDB.

10

Wie verarbeite ich Ausnahmen mit Fehlercodes?

Neben den AWS SDK für PHP benutzerdefinierten `Exception` Klassen hat jeder AWS Service-Client seine eigene Ausnahmeklasse, die von erbt. [AwsException](#) `AwsException`. Sie können spezifischere Fehlerarten festlegen, die mit den API-spezifischen Fehlern abgefangen werden sollen, die im `Errors`-Abschnitt der einzelnen Methoden aufgelistet sind.

Informationen zum Fehlercode sind mit [getAwsErrorCode\(\)](#) von verfügbar. `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;
    }
}
```

Glossar

API-Version

Services verfügen über eine oder mehrere API-Versionen. Welche Version Sie verwenden, schreibt vor, welche Vorgänge und Parameter gültig sind. API-Versionen sind wie ein Datum formatiert. Die neueste API-Version für Amazon S3 ist beispielsweise 2006-03-01. [Geben Sie eine Version an](#), wenn Sie ein Client-Objekt konfigurieren.

Client

Client-Objekte werden verwendet, um Operationen für einen Service auszuführen. Jeder im SDK unterstützte Service verfügt über ein entsprechendes Clientobjekt. Client-Objekte verfügen über Methoden, die one-to-one den Serviceoperationen entsprechen. Informationen [the section called "Einen Basis-Serviceclient erstellen"](#) zu den ersten Schritten mit der Verwendung von Clients finden Sie unter.

Befehl

Befehlsobjekte kapseln die Ausführung einer Operation ein. Wenn Sie einen Servicevorgang ausführen, wie im Abschnitt [Anfrage stellen](#) beschrieben, werden Sie sich nicht direkt mit Befehlsobjekten befassen. Auf Befehlsobjekte kann mit der Methode `getCommand()` eines Clients zugegriffen werden, um erweiterte Funktionen des SDK zu verwenden, z. B. gleichzeitige Anfragen und Stapelverarbeitung. Weitere Informationen finden Sie [im Handbuch zu Befehlsobjekten im Handbuch zu AWS SDK für PHP Version 3](#).

Handler

Ein Handler ist eine Funktion, die die eigentliche Transformation eines Befehls und einer Anfrage in ein Ergebnis durchführt. Ein Handler sendet typischerweise HTTP-Anfragen. Handler können mit Middleware konstruiert werden, um ihr Verhalten zu verbessern. Ein Handler ist eine Funktion, die eine `Aws\CommandInterface` und eine `Psr\Http\Message\RequestInterface` akzeptiert und ein `Promise` zurückgibt, das mit einer `Aws\ResultInterface` erfüllt oder mit einem `Aws\Exception\AwsException` Grund abgelehnt wird.

JMESPath

[JMESPath](#) ist eine Abfragesprache für JSON-ähnliche Daten. Die AWS SDK für PHP verwendet JMESPath Ausdrücke, um PHP-Datenstrukturen abzufragen. JMESPath Ausdrücke können über die `search($expression)` Methode direkt für `Aws\ResultPaginator` Objekte verwendet werden. `Aws\Result`

Middleware

Middleware ist eine spezielle High-Level-Funktion, die das Verhalten bei der Übertragung eines Befehls ergänzt und an einen "nächsten" Handler delegiert. Middleware-Funktionen akzeptieren eine `Aws\CommandInterface` und eine `Psr\Http\Message\RequestInterface` und geben ein `Promise` zurück, das mit einer `Aws\ResultInterface` erfüllt oder mit einem `Aws\Exception\AwsException`-Grund abgelehnt wird.

Operation

Bezieht sich auf eine einzelne Operation innerhalb der API eines Dienstes (z. B. `CreateTable` für DynamoDB, `RunInstances` für Amazon EC2). Im SDK werden Operationen ausgeführt, indem eine Methode mit demselben Namen im Clientobjekt des entsprechenden Service aufgerufen wird. Das Ausführen einer Operation umfasst das Vorbereiten und Senden einer HTTP-Anforderung an den Service und das Analysieren der Antwort. Dieser Vorgang zum Ausführen einer Operation wird vom SDK über Befehl Objekte abstrahiert.

Umbruch

Einige AWS Serviceoperationen sind paginiert und antworten mit verkürzten Ergebnissen. Der `ListObjects` Vorgang von Amazon S3 gibt beispielsweise nur bis zu 1000 Objekte gleichzeitig zurück. Für Operationen wie diese müssen nachfolgende Anfragen mit Token-Parametern (oder Marker-Parametern) gestellt werden, um alle Ergebnisse abzurufen. Paginatoren sind eine Funktion des SDK, die als Abstraktion für diesen Prozess dienen, um Entwicklern die Verwendung von paginierten Seiten zu erleichtern. APIs Sie werden über die `getPaginator()`-Methode des Client aufgerufen. Weitere Informationen finden Sie in den [Paginatoren im Handbuch](#) zu Version 3. AWS SDK für PHP

Promise

Ein `Promise` repräsentiert das Ergebnis einer asynchronen Operation. Der primäre Weg, mit einem `Promise` zu interagieren, ist eine Methode, die Callbacks registriert, um entweder den möglichen Wert eines `Promise`s zu erhalten oder den Grund, warum das `Promise` nicht erfüllt werden kann.

Region

Services werden in [einer oder mehreren geographischen Regionen](#) unterstützt. Die Dienste können endpoints/URLs in jeder Region unterschiedlich sein. Diese dienen dazu, die Datenlatenz in Ihren Anwendungen zu reduzieren. [Geben Sie eine Region an](#) beim Konfigurieren eines Clientobjekt, damit das SDK bestimmen kann, welcher Endpunkt mit dem Service verwendet werden soll.

SDK

Der Begriff „SDK“ kann sich auf die gesamte AWS SDK für PHP Bibliothek beziehen, bezieht sich aber auch auf die `Aws\Sdk` Klasse ([docs](#)), die als Factory für die Client-Objekte für jeden Dienst fungiert. Mit der Klasse `Sdk` können Sie auch einen Satz [globaler Konfigurationswerte](#) angeben, die auf alle von ihm erstellten Client-Objekte angewendet werden.

Service

Eine allgemeine Art, auf einen der AWS Dienste zu verweisen (z. B. Amazon S3, Amazon DynamoDB AWS OpsWorks usw.). Jeder Service hat ein entsprechendes Client-Objekt im SDK, das eine oder mehrere API-Versionen unterstützt. Jeder Service hat auch eine oder mehrere Operationen, die seine API bilden. Serviceleistungen werden in einer oder mehreren Regionen unterstützt.

Signatur

Beim Ausführen von Vorgängen verwendet das SDK Ihre Anmeldeinformationen zum Erstellen einer digitalen Signatur Ihrer Anfrage. Der Service überprüft dann die Signatur vor der Verarbeitung Ihrer Anfrage. Der Signierungsprozess wird vom SDK gekapselt und erfolgt automatisch mit den Anmeldeinformationen, die Sie für den Client konfigurieren.

Waiter

Waiter sind eine Funktion des SDK, die es einfacher macht, mit Operationen zu arbeiten, die den Status einer Ressource ändern und die von Natur aus letztendlich datenkonsistent oder asynchron sind. Beispielsweise sendet der Amazon DynamoDB `CreateTable` DynamoDB-Vorgang sofort eine Antwort zurück, aber die Tabelle ist möglicherweise erst nach einigen Sekunden zugriffsbereit. Durch Ausführen eines Waiters können Sie warten, bis eine Ressource in einen bestimmten Status übergeht, indem Sie den Status der Ressource ruhen lassen und abrufen. Der Zugriff auf Waiter erfolgt über die `MethodWaitUntil()` des Clients. Weitere Informationen finden Sie [im Handbuch Waiters in der AWS SDK für PHP Version 3](#).

Die neueste AWS Terminologie finden Sie im [AWS Glossar](#) in der Allgemeine AWS-Referenz

Dokumentverlauf

In den folgenden Tabellen werden die wichtigen Änderungen seit der letzten Version des AWS SDK für PHP Developer Guide beschrieben.

Die letzten Änderungen:

Änderung	Beschreibung	Datum
Aktualisierungen des Inhaltsverzeichnis	Das Inhaltsverzeichnis wurde überarbeitet, damit es besser mit anderen SDK-Handbüchern übereinstimmt.	28. Juli 2025
Schutz der Datenintegrität mit Prüfsummen	Der Inhalt wurde mit Einzelheiten zur automatischen Prüfsummenberechnung aktualisiert.	16. Januar 2025
Überarbeitungen des Themas „Anmeldeinformationen“	Das Thema wurde neu organisiert. Weitere Informationen zur Anbieterkette für Standardanmeldedaten finden Sie hier.	10. Januar 2025
Migration des Amazon S3 S3-Verschlüsselungsclients V3	Thema zur Migration des Amazon S3 S3-Verschlüsselungsclients von V2 zu V3 hinzugefügt	4. Dezember 2024
Mehrteilige Amazon S3 S3-Uploads	Dokumentieren Sie das Array 'params', das zur Konfiguration der Unterbefehle von und verwendet werden kann <code>ObjectUploader</code> <code>MultipartUploader</code>	6. November 2024

Objekt-Uploader	Erläutern Sie die Verwendung der <code>ObjectUploader</code> für S3-Uploads verfügbaren Callbacks	11. Oktober 2024
Amazon S3 S3-Bucket-Namen aktualisieren	Die S3-Bucket-Namen wurden im gesamten Handbuch aktualisiert.	30. September 2024
EventBridge Globale Amazon-Endpunkte	Fügen Sie ein Codebeispiel hinzu, das zeigt, wie Sie EventBridge globale Amazon-Endpunkte verwenden	22. Dezember 2023
AWS Allgemeine Laufzeit (AWS CRT)	Fügen Sie ein Thema hinzu, das die Verwendung der AWS Common Runtime (AWS CRT) durch das SDK for PHP behandelt.	17. November 2023
StreamWrapper mkdir () aktualisiert	Fügen Sie Informationen über die Arbeit mit Buckets und Ordnerobjekten hinzu, indem Sie <code>mkdir()</code>	2. November 2022
Erstellung von Serviceclients	Aktualisieren Sie Codefragmente, indem Sie den Parameter „Version“ entfernen, da der Parameter „latest“ der Standard ist.	31. August 2023
Grundlegende Client-Erstellung	Aktualisieren Sie Codefragmente, indem Sie den Parameter „Version“ entfernen, da der Parameter „Latest“ der Standard ist.	31. August 2023

Inhaltsverzeichnis	Das Inhaltsverzeichnis wurde aktualisiert, um den Zugriff auf Codebeispiele zu erleichtern.	01. Juni 2023
Aktualisierungen der bewährten Methoden für IAM	Aktualisierung des Leitfadens zur Ausrichtung an bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden in IAM . Aktualisierungen für Erste Schritte.	20. Mai 2023
Mehrteilige Amazon S3 S3-Uploads	Inklusive Konfigurationsinformationen für synchrone Uploads. Die <code>add_content_md5</code> Upload-Option für asynchrone Uploads wurde hinzugefügt.	13. April 2023
Amazon S3 S3-Verzeichnisübertragungen	Die <code>add_content_md5</code> Übertragungsoption wurde hinzugefügt.	13. April 2023
Referenzinformationen	Es wurden mehrere Links zu relevanten Detailinhalten im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch hinzugefügt. Die Formatierung des Handbuchs wurde aktualisiert.	14. September 2022

Allgemeine Säuberung	Es wurden Verweise auf das Referenzhandbuch AWS SDKs und das Tools-Referenzhandbuch hinzugefügt. Die AWS Key Management Service Abschnitte wurden aktualisiert, um Aktualisierungen der Terminologie widerzuspiegeln.	23. August 2022
Mit AWS Diensten arbeiten	Enthalten sind Listen der Codebeispiele, die auf verfügbar sind GitHub.	1. April 2022
SDK-Metriken aktivieren	Informationen zur Aktivierung von SDK-Metriken, die am 20. Dezember 2021 veraltet waren, wurden entfernt.	27. Januar 2022
Migration des Amazon S3 S3-Verschlüsselungsclients	Thema zur Migration von Amazon S3 S3-Verschlüsselungsclients hinzugefügt	7. August 2020

Ältere Änderungen:

Änderungen	Beschreibung	Datum der Veröffentlichung
Beispiele für Secrets Manager	Hinzufügen weiterer Servicebeispiele	27. März 2019
Endpunkterkennung	Konfiguration der Endpunkterkennung	15. Februar 2019
Amazon CloudFront	Hinzufügen weiterer Servicebeispiele	25. Januar 2019
Servicefunktionen	SDK-Metriken	11. Januar 2018

Änderungen	Beschreibung	Datum der Veröffentlichung
Amazon Kinesis, Amazon SNS	Hinzufügen weiterer Servicebeispiele	14. Dezember 2018
Amazon-SES-Beispiele	Hinzufügen weiterer Servicebeispiele	5. Oktober 2018
AWS KMS Beispiele	Hinzufügen weiterer Servicebeispiele	8. August 2018
Anmeldeinformationen	Klarstellen und Vereinfachen des Leitfadens zu Anmeldeinformationen	30. Juni 2018
MediaConvert Beispiele	Hinzufügen weiterer Servicebeispiele	15. Juni 2018
Neues Web-Layout	Die Dokumentation wurde auf AWS Stil umgestellt	9. Mai 2018
Amazon-S3-Verschlüsselung	Clientseitige Verschlüsselung	17. November 2017
Amazon S3, Amazon SQS	Hinzufügen weiterer Servicebeispiele	26. März 2017
Amazon S3, IAM, Amazon EC2	Hinzufügen weiterer Servicebeispiele	17. März 2017
Hinzufügen von Anmeldeinformationen	Fügt Unterstützung für AssumeRole und ini hinzu	17. Januar 2017
S3-Beispiele	S3 Multi-Region und vorsignierte Beiträge	18. März 2016
OpenSearch Service und Amazon CloudSearch	Hinzufügen weiterer Servicebeispiele	28. Dezember 2015
Befehlszeile	Hinzufügen von Befehlsparametern	13. August 2015

Änderungen	Beschreibung	Datum der Veröffentlichung
Servicefunktionen	Fügt Servicefunktionen für S3 hinzu und AWS	30. April 2015
Neue SDK-Version	Version 3 der AWS SDK für PHP veröffentlichten.	26. Mai 2015

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.