Entwicklerhandbuch

AWS SDK for Ruby

AWS SDK for Ruby: Entwicklerhandbuch

Table of Contents

Was ist das AWS SDK for Ruby?

Willkommen im AWS SDK for Ruby Developer Guide. Das AWS SDK for Ruby bietet Unterstützungsbibliotheken für fast alle AWS-Services, einschließlich Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2) und Amazon DynamoDB.

Das AWS SDK for Ruby Developer Guide enthält Informationen zur Installation, Einrichtung und Verwendung des AWS SDK for Ruby, um Ruby-Anwendungen zu erstellen, die AWS-Services

Erste Schritte mit dem AWS SDK for Ruby

Zusätzliche Dokumentation und Ressourcen

Weitere Ressourcen für AWS SDK for Ruby Ruby-Entwickler finden Sie im Folgenden:

- AWS SDKs Referenzhandbuch für Tools und Tools — Enthält Einstellungen, Funktionen und andere grundlegende Konzepte, die unter AWS SDKs
- AWS SDK für Ruby API-Referenz — Version 3
- AWS Codebeispiele: Repository auf GitHub
- RubyGems.org — Die neueste Version des SDK ist in dienstspezifische Gems modularisiert, die hier verfügbar sind
  - Unterstützte Dienste — Listet alle Gems auf, die das AWS SDK for Ruby unterstützt
- AWS SDK for Ruby Ruby-Quellcode auf GitHub:
  - Quelle und README
  - Änderungsprotokolle unter jedem Gem
  - Migrieren von v2 zu v3
  - Problembereiche
  - Hinweise zu Core-Upgrades
- Entwickler-Blog

Bereitstellung in der Cloud AWS

Sie können AWS-Services diese AWS Elastic Beanstalk und verwenden AWS CodeDeploy , um Ihre Anwendung in der AWS Cloud bereitzustellen. Informationen zur Bereitstellung von Ruby-

Anwendungen mit Elastic Beanstalk finden Sie unter Deployment Elastic Beanstalk Applications in Ruby Using EB CLI and Git im Developer Guide. AWS Elastic Beanstalk Einen Überblick über die AWS Bereitstellungsservices finden Sie unter Überblick über die Bereitstellungsoptionen auf. AWS

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 AWS SDKs und im Tools-Referenzhandbuch:

- AWS SDKs und Richtlinien zur Wartung von Tools
- AWS SDKs Matrix zur Support von Versionen und Tools

Erste Schritte mit dem AWS SDK for Ruby

Erfahren Sie, wie Sie das SDK installieren, einrichten und verwenden, um eine Ruby-Anwendung für den programmgesteuerten Zugriff auf eine AWS Ressource zu erstellen.

Themen

Authentifizierung mit AWS AWS SDK for Ruby

Sie müssen festlegen, wie sich Ihr Code AWS bei der Entwicklung mit AWS-Services 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.

Verwenden von Anmeldeinformationen für die Konsole

Für die lokale Entwicklung empfehlen wir neuen Benutzern, ihre vorhandenen Anmeldedaten für die AWS Management Console für den programmgesteuerten Zugriff auf Dienste zu AWS verwenden. AWS Generiert nach der browserbasierten Authentifizierung temporäre Anmeldeinformationen, die mit lokalen Entwicklungstools wie der AWS Befehlszeilenschnittstelle (AWS CLI) und dem AWS SDK for Ruby funktionieren.

Wenn Sie diese Methode wählen, folgen Sie den Anweisungen zur Anmeldung für die AWS lokale Entwicklung mit Konsolenanmeldedaten über die AWS CLI.

Das AWS SDK for Ruby benötigt keine zusätzlichen Gems (wie`aws-sdk-signin`), die zu Ihrer Anwendung hinzugefügt werden müssen, um die Anmeldung mit Konsolenanmeldedaten zu verwenden.

Verwenden der IAM Identity Center-Authentifizierung

Wenn Sie sich für diese Methode entscheiden, führen Sie das Verfahren für die IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch durch. Danach sollte Ihre Umgebung die folgenden Elemente enthalten:

- Die AWS CLI, mit der Sie eine AWS Access-Portal-Sitzung starten, bevor Sie Ihre Anwendung ausführen.
- Eine gemeinsam genutzte AWS`config` Datei mit einem `[default]` Profil mit einer Reihe von Konfigurationswerten, auf die vom SDK aus verwiesen werden 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 legt die `region`Einstellung fest. Dies legt die Standardeinstellung AWS-Region fest, die das SDK für AWS Anfragen verwendet. Diese Region wird für SDK-Dienstanforderungen verwendet, für die keine zu verwendende Region angegeben ist.
- 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 genannten `sso-session`Abschnitt. 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
```

Das AWS SDK for Ruby benötigt keine zusätzlichen Gems (wie `aws-sdk-sso` und`aws-sdk-ssooidc`), die zu Ihrer Anwendung hinzugefügt werden müssen, um die IAM Identity Center-Authentifizierung zu verwenden.

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.

Weitere Authentifizierungsinformationen

Menschliche Benutzer, auch bekannt als menschliche Identitäten, sind die Personen, Administratoren, Entwickler, Betreiber und Verbraucher Ihrer Anwendungen. Sie benötigen eine Identität, um auf Ihre AWS Umgebungen und Anwendungen zugreifen zu können. Menschliche Benutzer, die Mitglieder Ihres Unternehmens sind, also Sie, der Entwickler, werden als Personalidentitäten bezeichnet.

Verwenden Sie beim Zugriff AWS temporäre Anmeldeinformationen. Sie können einen Identitätsanbieter für Ihre menschlichen Benutzer verwenden, um Verbundzugriff auf AWS Konten zu ermöglichen, indem Sie Rollen übernehmen, die temporäre Anmeldeinformationen bereitstellen. Für eine zentralisierte Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center (IAM Identity Center) zu verwenden, um den Zugriff auf Ihre Konten und die Berechtigungen innerhalb dieser Konten zu verwalten. Weitere Alternativen finden Sie im Folgenden:

- Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Informationen zum Erstellen kurzfristiger AWS Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.
- Weitere Informationen über die Anmeldeinformationsanbieterkette des AWS SDK for Ruby und darüber, wie verschiedene Authentifizierungsmethoden vom SDK automatisch nacheinander versucht werden, finden Sie unterKette von Anbietern von Anmeldeinformationen.
- Informationen zu den Konfigurationseinstellungen des AWS SDK-Anmeldeinformationsanbieters finden Sie unter Standardisierte Anmeldeinformationsanbieter im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Das AWS SDK for Ruby installieren

Dieser Abschnitt enthält Voraussetzungen und Installationsanweisungen für das AWS SDK for Ruby.

Voraussetzungen

Bevor Sie das AWS SDK for Ruby verwenden, müssen Sie sich mit AWS authentifizieren. Informationen zum Einrichten der Authentifizierung finden Sie unterAuthentifizierung mit AWS AWS SDK for Ruby.

Installieren des SDK

Sie können das AWS SDK for Ruby wie jedes Ruby-Gem installieren. Die Edelsteine sind erhältlich unter RubyGems. Das AWS SDK for Ruby ist modular konzipiert und wird durch getrennt AWS-Service. Die Installation des gesamten `aws-sdk` Edelsteins ist umfangreich und kann über eine Stunde dauern.

Wir empfehlen, die Edelsteine nur für das zu installieren, AWS-Services was Sie verwenden. Diese sind wie benannt `aws-sdk-service_abbreviation` und die vollständige Liste finden Sie in der Tabelle Unterstützte Dienste der README-Datei AWS SDK for Ruby. Zum Beispiel ist das Gem für die Schnittstelle mit dem Amazon S3 S3-Service direkt verfügbar unter `aws-sdk-s3`.

Ruby-Versionsmanager

Anstatt System-Ruby zu verwenden, empfehlen wir, einen Ruby-Versionsmanager wie den folgenden zu verwenden:

- RVM
- Chruby
- rbenv

Wenn Sie beispielsweise ein Amazon Linux 2-Betriebssystem verwenden, können Sie die folgenden Befehle verwenden, um RVM zu aktualisieren, die verfügbaren Ruby-Versionen aufzulisten und dann die Version auszuwählen, die Sie für die Entwicklung mit dem AWS SDK for Ruby verwenden möchten. Die mindestens erforderliche Ruby-Version ist 2.5.

```
$ rvm get head
$ rvm list known
$ rvm install ruby-3.1.3
$ rvm --default use 3.1.3
```

Bundler

Wenn Sie Bundler verwenden, installieren Sie mit den folgenden Befehlen das AWS SDK for Ruby Gem für Amazon S3:

1. Installieren Sie Bundler und erstellen Sie: `Gemfile`

   ```
   $ gem install bundler
   ```

```
$ bundle init
```

2. Öffnen Sie das erstellte `Gemfile` und fügen Sie eine `gem` Zeile für jedes AWS Service-Gem hinzu, das Ihr Code verwenden wird. Um dem Amazon S3 S3-Beispiel zu folgen, fügen Sie am Ende der Datei die folgende Zeile hinzu:

```
gem "aws-sdk-s3"
```

3. Speichern Sie das Gemfile.
4. Installieren Sie die Abhängigkeiten, die in Ihrem`Gemfile`:

```
$ bundle install
```

Erstellen einer einfachen Anwendung mit dem AWS SDK for Ruby

Begrüßen Sie Amazon S3 mithilfe des AWS SDK for Ruby. Im folgenden Beispiel wird eine Liste Ihrer Amazon S3 S3-Buckets angezeigt.

Den Code schreiben

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
```

```
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS Das SDK for Ruby ist modular konzipiert und wird durch getrennt AWS-Service. Nach der Installation des Gems importiert die `require` Anweisung oben in Ihrer Ruby-Quelldatei die AWS SDK-Klassen und -Methoden für den Amazon S3 S3-Service. Eine vollständige Liste der verfügbaren AWS Service Gems finden Sie in der Tabelle Unterstützte Dienste der README-Datei AWS SDK for Ruby.

```
require 'aws-sdk-s3'
```

Das Programm wird ausgeführt

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

```
ruby [source filename] [arguments...]
```

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

```
$ ruby hello-s3.rb
```

Hinweis für Benutzer von Windows

Wenn Sie SSL-Zertifikate unter Windows verwenden und Ihren Ruby-Code ausführen, wird möglicherweise ein Fehler ähnlich dem folgenden angezeigt.

```
C:\Ruby>ruby buckets.rb
C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect': SSL_connect returned=1
 errno=0 state=SSLv3 read server certificate B: certificate verify failed
 (Seahorse::Client::NetworkingError)
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `block in connect'

        from C:/Ruby200-x64/lib/ruby/2.0.0/timeout.rb:66:in `timeout'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:862:in `do_start'
        from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:857:in `start'
...
```

Um dieses Problem zu beheben, fügen Sie Ihrer Ruby-Quelldatei die folgende Zeile hinzu, und zwar irgendwo vor Ihrem ersten AWS Aufruf.

```
Aws.use_bundled_cert!
```

Wenn Sie nur das `aws-sdk-s3` Gem in Ihrem Ruby-Programm verwenden und das mitgelieferte Zertifikat verwenden möchten, müssen Sie auch das `aws-sdk-core` Gem hinzufügen.

Nächste Schritte

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

Konfiguration von Service-Clients im AWS SDK for Ruby

Für den programmgesteuerten Zugriff AWS-Services verwendet das AWS SDK for Ruby jeweils eine Client-Klasse. AWS-Service Wenn Ihre Anwendung beispielsweise auf Amazon zugreifen muss EC2, erstellt Ihre Anwendung ein EC2 Amazon-Client-Objekt als Schnittstelle zu diesem Service. Anschließend verwenden Sie den Service-Client, um Anfragen an dieses zu stellen AWS-Service.

Um eine Anfrage an einen zu stellen AWS-Service, müssen Sie zunächst einen Service-Client erstellen. Für jeden Code, den AWS-Service Sie verwenden, gibt es ein eigenes Gem und einen eigenen Typ, mit dem Sie interagieren können. Der Client stellt für jeden API-Vorgang, der vom Dienst verfügbar gemacht wird, eine Methode zur Verfügung.

Es gibt viele alternative Möglichkeiten, das SDK-Verhalten zu konfigurieren, aber letztendlich hat alles mit dem Verhalten von Service-Clients zu tun. Jede Konfiguration hat keine Wirkung, bis ein aus ihnen erstellter Service-Client verwendet wird.

Sie müssen festlegen, wie sich Ihr Code authentifiziert AWS , wenn Sie mit AWS-Services entwickeln. Sie müssen auch festlegen, was AWS-Region Sie verwenden möchten.

Das AWS SDKs Referenzhandbuch zu Tools enthält außerdem Einstellungen, Funktionen und andere grundlegende Konzepte, die in vielen der AWS SDKs Tools üblich sind.

Themen

Die Dateien „Gemeinsam genutzt“ `config` und „`credentials`Dateien“ können für Konfigurationseinstellungen verwendet werden. Alle AWS SDK-Einstellungen finden Sie in der Einstellungsreferenz im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Verschiedene Profile können zum Speichern verschiedener Konfigurationen verwendet werden. Um das aktive Profil anzugeben, das das SDK lädt, können Sie die `AWS_PROFILE` Umgebungsvariable oder die `profile` Option von verwenden`Aws.config`.

Vorrang der Einstellungen

In globalen Einstellungen werden Funktionen, Anbieter von Anmeldeinformationen und andere Funktionen konfiguriert, die von den meisten unterstützt werden SDKs und weitreichende Auswirkungen auf alle haben. AWS-Services Alle AWS SDKs haben eine Reihe von Orten (oder Quellen), die sie überprüfen, um einen Wert für globale Einstellungen zu finden. Nicht alle Einstellungen sind in allen Quellen verfügbar. Im Folgenden finden Sie die Einstellung „Priorität bei der Suche“:

1. Jede explizite Einstellung, die im Code oder auf einem Service-Client selbst festgelegt ist, hat Vorrang vor allen anderen Einstellungen.
   a. Alle Parameter, die direkt an einen Client-Konstruktor übergeben werden, haben höchste Priorität.
   b. `Aws.config`wird auf globale oder dienstspezifische Einstellungen überprüft.
2. Die Umgebungsvariable wird geprüft.
3. Die gemeinsam genutzte AWS `credentials` Datei wird überprüft.
4. Die gemeinsam genutzte AWS `config` Datei ist geprüft.
5. Jeder Standardwert, der vom AWS SDK für den Ruby-Quellcode selbst bereitgestellt wird, wird zuletzt verwendet.

Externes Konfigurieren des AWS SDK for Ruby Ruby-Serviceclients

Viele Konfigurationseinstellungen können außerhalb Ihres Codes bearbeitet werden. Wenn die Konfiguration extern vorgenommen wird, wird die Konfiguration auf alle Ihre Anwendungen angewendet. Die meisten Konfigurationseinstellungen können entweder als Umgebungsvariablen oder in einer separaten gemeinsam genutzten AWS `config` Datei festgelegt werden. Die gemeinsam genutzte `config` Datei kann separate Einstellungssätze, sogenannte Profile, enthalten, um unterschiedliche Konfigurationen für verschiedene Umgebungen oder Tests bereitzustellen.

Umgebungsvariablen und gemeinsam genutzte `config` Dateieinstellungen sind standardisiert und werden von allen AWS SDKs Tools gemeinsam genutzt, um konsistente Funktionen in verschiedenen Programmiersprachen und Anwendungen zu unterstützen.

Weitere Informationen zur Konfiguration Ihrer Anwendung mit diesen Methoden AWS SDKs sowie Einzelheiten zu den einzelnen SDK-übergreifenden Einstellungen finden Sie im Referenzhandbuch und im Tools-Referenzhandbuch. 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.

Um eine Anfrage an zu stellen AWS-Service, instanziieren Sie zunächst einen Client für diesen Dienst. Sie können allgemeine Einstellungen für Service-Clients wie Timeouts und den HTTP-Client konfigurieren und die Konfiguration erneut versuchen.

Jeder Dienstclient benötigt einen Anmeldeinformationsanbieter AWS-Region und einen Anmeldeinformationsanbieter. Das SDK verwendet diese Werte, um Anfragen an die richtige Region für Ihre Ressourcen zu senden und Anfragen mit den richtigen Anmeldeinformationen zu signieren. Sie können diese Werte programmgesteuert im Code angeben oder sie automatisch aus der Umgebung laden lassen.

Das SDK verfügt über eine Reihe von Stellen (oder Quellen), die überprüft werden, um einen Wert für Konfigurationseinstellungen zu finden.

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 vom SDK-Quellcode selbst bereitgestellte Standardwert wird zuletzt verwendet.
   - Für einige Eigenschaften, 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.

AWS SDK for Ruby Ruby-Umgebungsvariablen

Neben den SDK-übergreifenden Umgebungsvariablen, die von den meisten unterstützt werden AWS SDKs, unterstützt das AWS SDK for Ruby einige einzigartige Variablen:

`AWS_SDK_CONFIG_OPT_OUT`

Wenn die Umgebungsvariable AWS SDK for Ruby gesetzt `AWS_SDK_CONFIG_OPT_OUT` ist, wird die gemeinsam genutzte AWS `config` Datei `~/.aws/config`, normalerweise at, für keine Konfigurationswerte verwendet.

`AMAZON_REGION`

Eine alternative Umgebungsvariable `AWS_REGION` zur Einstellung von AWS-Region. Dieser Wert wird nur geprüft, wenn `AWS_REGION` er nicht verwendet wird.

AWS SDK for Ruby Ruby-Serviceclients im Code konfigurieren

Wenn die Konfiguration direkt im Code abgewickelt wird, ist der Konfigurationsbereich auf die Anwendung beschränkt, die diesen Code verwendet. Innerhalb dieser Anwendung gibt es Optionen für die globale Konfiguration aller Service-Clients, die Konfiguration für alle Clients eines bestimmten AWS-Service Typs oder die Konfiguration für eine bestimmte Service-Client-Instanz.

`Aws.config`

Um in Ihrem Code eine globale Konfiguration für alle AWS Klassen bereitzustellen, verwenden Sie `Aws.config`die im `aws-sdk-core` Gem verfügbare Konfiguration.

`Aws.config`unterstützt zwei Syntaxen für unterschiedliche Zwecke. Globale Einstellungen können entweder für alle AWS-Services oder für einen bestimmten Dienst angewendet werden. Die vollständige Liste der unterstützten Einstellungen finden Sie `Client` `Options`in der AWS SDK für Ruby API-Referenz.

Globale Einstellungen über `Aws.config`

Verwenden Sie die folgende Syntax`Aws.config`, um dienstunabhängige Einstellungen festzulegen:

```
Aws.config[:<global setting name>] = <value>
```

Diese Einstellungen werden mit allen erstellten Service-Clients zusammengeführt.

Beispiel für eine globale Einstellung:

```
Aws.config[:region] = 'us-west-2'
```

Wenn Sie versuchen, einen Einstellungsnamen zu verwenden, der nicht global unterstützt wird, wird ein Fehler ausgelöst, wenn Sie versuchen, eine Instanz eines Diensttyps zu erstellen, der ihn nicht unterstützt. Verwenden Sie in diesem Fall stattdessen eine dienstspezifische Syntax.

Servicespezifische Einstellungen über **`Aws.config`**

Verwenden Sie die folgende Syntax`Aws.config`, um dienstspezifische Einstellungen festzulegen:

```
Aws.config[:<service identifier>] = { <global setting name>: <value> }
```

Diese Einstellungen werden in allen erstellten Dienstclients dieses Diensttyps zusammengeführt.

Beispiel für eine Einstellung, die nur für Amazon S3 gilt:

```
Aws.config[:s3] = { force_path_style: true }
```

Der *<service identifier>* kann identifiziert werden, indem man sich den Namen des entsprechenden AWS SDK for Ruby Ruby-Gems ansieht und das Suffix verwendet, das auf "`aws-sdk-`“ folgt. Beispiel:

- Für lautet `aws-sdk-s3` die Zeichenfolge für die Dienstkennung "`s3`“.
- Für `aws-sdk-ecs` lautet die Dienstkennungszeichenfolge "`ecs`“.

Einstellung des AWS-Region für das AWS SDK for Ruby

Sie können auf diejenigen zugreifen AWS-Services , die in einem bestimmten geografischen Gebiet tätig sind, indem Sie AWS-Regionen. Dies kann sowohl aus Gründen der Redundanz als auch dafür nützlich sein, dass Ihre Daten und Anwendungen in der Nähe ausgeführt werden, wo Sie und Ihre Benutzer darauf zugreifen.

> **Important**
>
> Die meisten Ressourcen befinden sich in einer bestimmten Region, AWS-Region und Sie müssen die richtige Region für die Ressource angeben, wenn Sie das SDK verwenden.

Sie müssen einen Standard AWS-Region für das SDK for Ruby festlegen, das für AWS Anfragen verwendet werden soll. Dieser Standard wird für alle Aufrufe von SDK-Dienstmethoden verwendet, die nicht mit einer Region angegeben sind.

Weitere Informationen zu dieser `region` Einstellung finden Sie AWS-Regionim Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. Dazu gehören auch Beispiele zur Festlegung der Standardregion mithilfe der gemeinsam genutzten AWS `config` Datei- oder Umgebungsvariablen.

Reihenfolge der Suche nach Regionen zur Auflösung

Sie müssen eine Region festlegen, wenn Sie die meisten verwenden AWS-Services. Das AWS SDK for Ruby sucht in der folgenden Reihenfolge nach einer Region:

1. Einstellung der Region in einem Client- oder Ressourcenobjekt
2. Einstellen der Region mithilfe von `Aws.config`
3. Einstellen der Region mithilfe von Umgebungsvariablen
4. Einstellen der Region mithilfe der gemeinsam genutzten `config` Datei

Wie lege ich die Region fest

In diesem Abschnitt werden verschiedene Möglichkeiten zur Festlegung einer Region beschrieben, beginnend mit der gängigsten Methode.

Einstellen der Region mithilfe der gemeinsam genutzten **`config`** Datei

Legen Sie die Region fest, indem Sie die `region` Variable in der gemeinsam genutzten AWS `config` Datei festlegen. Weitere Informationen zur gemeinsam genutzten `config` Datei finden Sie unter Dateien mit gemeinsam genutzten Konfigurationen und Anmeldeinformationen im AWS SDKs Referenzhandbuch zu Tools.

Beispiel für die Einstellung dieses Werts in der `config` Datei:

```
[default]
region = us-west-2
```

Die gemeinsam genutzte `config` Datei wird nicht überprüft, wenn die Umgebungsvariable gesetzt `AWS_SDK_CONFIG_OPT_OUT` ist.

Einstellung der Region mithilfe von Umgebungsvariablen

Legen Sie die Region fest, indem Sie die `AWS_REGION` Umgebungsvariable festlegen.

Verwenden Sie den `export` Befehl, um diese Variable auf UNIX-basierten Systemen wie Linux oder macOS festzulegen. Im folgenden Beispiel wird die Region auf festgelegt. `us-west-2`

```
export AWS_REGION=us-west-2
```

Verwenden Sie den `set`-Befehl, um diese Variable unter Windows zu definieren. Im folgenden Beispiel wird die Region auf festgelegt`us-west-2`.

```
set AWS_REGION=us-west-2
```

Einstellung der Region mit **`Aws.config`**

Legen Sie die Region fest, indem Sie dem `Aws.config` Hash einen `region` Wert hinzufügen. Im folgenden Beispiel wird der `Aws.config` Hash aktualisiert, sodass er die `us-west-1` Region verwendet.

```
Aws.config.update({region: 'us-west-1'})
```

Alle Clients oder Ressourcen, die Sie später erstellen, sind an diese Region gebunden.

Einstellung der Region in einem Client- oder Ressourcenobjekt

Legen Sie die Region fest, wenn Sie einen AWS Client oder eine Ressource erstellen. Im folgenden Beispiel wird ein Amazon S3 S3-Ressourcenobjekt in der `us-west-1` Region erstellt. Wählen Sie die richtige Region für Ihre AWS Ressourcen. Ein Service-Client-Objekt ist unveränderlich. Sie müssen also für jeden Service, an den Sie Anfragen stellen, und für Anfragen an denselben Service mit einer anderen Konfiguration einen neuen Client erstellen.

```
s3 = Aws::S3::Resource.new(region: 'us-west-1')
```

Verwenden des AWS SDK for Ruby Ruby-Anmeldeinformationsanbieter

Alle Anfragen an AWS müssen kryptografisch signiert werden, wobei die Anmeldeinformationen verwendet werden müssen, die von ausgestellt wurden. AWS Zur Laufzeit ruft das SDK Konfigurationswerte für Anmeldeinformationen ab, indem es mehrere Speicherorte überprüft.

Die Authentifizierung mit AWS kann außerhalb Ihrer Codebasis erfolgen. Viele Authentifizierungsmethoden können vom SDK mithilfe der Credential Provider-Kette automatisch erkannt, verwendet und aktualisiert werden.

Anleitungen zu den ersten Schritten mit der AWS Authentifizierung für Ihr Projekt finden Sie unter Authentifizierung und Zugriff im AWS SDKs Referenzhandbuch zu Tools.

Kette von Anbietern von Anmeldeinformationen

Alle SDKs haben eine Reihe von Stellen (oder Quellen), die sie überprüfen, um gültige Anmeldeinformationen zu erhalten, mit denen eine Anfrage an AWS-Service einen gestellt werden kann. Nachdem gültige Anmeldeinformationen gefunden wurden, wird die Suche beendet. Diese systematische Suche wird als Standardanbieterkette für Anmeldeinformationen bezeichnet.

> **Note**
>
> Wenn Sie für den Einstieg den empfohlenen Ansatz für neue Benutzer befolgt haben, haben Sie sich währenddessen mit Anmeldedaten für die Konsole authentifiziert. Authentifizierung mit AWS AWS SDK for Ruby Andere Authentifizierungsmethoden sind in verschiedenen Situationen nützlich. Um Sicherheitsrisiken zu vermeiden, empfehlen wir, immer kurzfristige Anmeldeinformationen zu verwenden. Informationen zu anderen Authentifizierungsmethoden finden Sie unter Authentifizierung und Zugriff im AWS SDKs Referenzhandbuch zu Tools.

Für jeden Schritt in der Kette gibt es unterschiedliche Möglichkeiten, die Werte festzulegen. Das Setzen von Werten direkt im Code hat immer Vorrang, gefolgt von der Einstellung als Umgebungsvariablen und dann in der gemeinsam genutzten AWS `config` Datei.

Das Referenzhandbuch AWS SDKs und Tools enthält Informationen zu den SDK-Konfigurationseinstellungen, die von all AWS SDKs und dem AWS CLI verwendet werden. Weitere Informationen zur Konfiguration des SDK mithilfe der gemeinsam genutzten AWS `config` Datei

finden Sie unter Gemeinsam genutzte Konfigurations - und Anmeldeinformationsdateien. Weitere Informationen zur Konfiguration des SDK durch das Setzen von Umgebungsvariablen finden Sie unter Unterstützung von Umgebungsvariablen.

Zur Authentifizierung überprüft das AWS SDK for Ruby die Anmeldeinformationsanbieter in der Reihenfolge, die in der folgenden Tabelle aufgeführt ist. AWS

Anbieter von Anmeldeinformationen nach Priorität	AWS SDKs und Referenzhandbuch für Tools	AWS SDK für Ruby API Reference
AWS Zugangsschlüssel (temporäre und langfristige Anmeldeinformationen)	AWS Zugriffstasten	`Aws::Credentials` `Aws::SharedCredentials`
Web-Identitätstoken von AWS-Security-Token-Service (AWS STS)	Nehmen Sie die Rolle des Anbieters von Anmeldeinformationen an Verwenden von `role_arn`, `role_session_name`, und `web_identity_token_file`	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center. In diesem Handbuch finden Sie weitere InformationenAuthentifizierung mit AWS AWS SDK for Ruby.	Anbieter von IAM Identity Center-Anmeldeinformationen	`Aws::SSOCredentials`
Vertrauenswürdiger Entitätsanbieter (z. B.`AWS_ROLE_ARN`). In diesem Handbuch finden Sie weitere Informationen unterEin AWS STS Zugriffstoken erstellen.	Nehmen Sie die Rolle des Anbieters von Anmeldeinformationen an Verwenden von `role_arn` und `role_session_name`	`Aws::AssumeRoleCredentials`
Anbieter für Anmeldeinformationen	Anbieter von Anmeldeinformationen	`Aws::LoginCredentials`

Anbieter von Anmeldeinformationen nach Priorität	AWS SDKs und Referenzhandbuch für Tools	AWS SDK für Ruby API Reference
Anbieter für Prozessanmeldeinformationen	Anbieter für Anmeldeinformationen verarbeiten	`Aws::ProcessCredentials`
Anmeldeinformationen für Amazon Elastic Container Service (Amazon ECS)	Anbieter von Container-Anmeldeinformationen	`Aws::ECSCredentials`
Anmeldeinformationen für das Amazon Elastic Compute Cloud (Amazon EC2) - Instanzprofil (IMDS-Anmeldeinformationsanbieter)	Anbieter von IMDS-Anmeldeinformationen	`Aws::InstanceProfileCredentials`

Wenn die Umgebungsvariable AWS SDK for Ruby gesetzt `AWS_SDK_CONFIG_OPT_OUT` ist, wird die gemeinsam genutzte AWS `config` Datei `~/.aws/config`, normalerweise unter, nicht auf Anmeldeinformationen analysiert.

Ein AWS STS Zugriffstoken erstellen

Die Übernahme einer Rolle beinhaltet die Verwendung einer Reihe temporärer Sicherheitsanmeldedaten, mit denen Sie auf AWS Ressourcen zugreifen können, auf die Sie normalerweise keinen Zugriff haben. Diese temporären Anmeldeinformationen bestehen aus einer Zugriffsschlüssel-ID, einem geheimen Zugriffsschlüssel und einem Sicherheits-Token. Sie können die `Aws::AssumeRoleCredentials`Methode verwenden, um ein Zugriffstoken AWS -Security-Token-Service (AWS STS) zu erstellen.

Im folgenden Beispiel `linked::account::arn` wird ein Zugriffstoken verwendet, um ein Amazon S3 S3-Client-Objekt zu erstellen, wobei der Amazon-Ressourcenname (ARN) der Rolle, die übernommen werden soll, und ein Bezeichner für die angenommene Rollensitzung `session-name` ist.

```
role_credentials = Aws::AssumeRoleCredentials.new(
  client: Aws::STS::Client.new,
  role_arn: "linked::account::arn",
  role_session_name: "session-name"
```

```
)

s3 = Aws::S3::Client.new(credentials: role_credentials)
```

Weitere Informationen zur Einstellung `role_arn` oder `role_session_name` Einstellung dieser Einstellungen stattdessen mithilfe der gemeinsam genutzten AWS `config` Datei finden Sie unter Assume role Credential Provider im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Konfiguration von Wiederholungsversuchen im AWS SDK for Ruby

Das AWS SDK for Ruby bietet ein standardmäßiges Wiederholungsverhalten für Serviceanfragen und anpassbare Konfigurationsoptionen. Ruft auf, um AWS-Services gelegentlich unerwartete Ausnahmen zurückzugeben. Bestimmte Arten von Fehlern, wie Drosselungen oder vorübergehende Fehler, können erfolgreich sein, wenn der Aufruf erneut versucht wird.

Das Wiederholungsverhalten kann global mithilfe von Umgebungsvariablen oder Einstellungen in der gemeinsam genutzten Datei konfiguriert werden. AWS `config` Informationen zu diesem Ansatz finden Sie unter Verhalten bei Wiederholungsversuchen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. Es enthält auch detaillierte Informationen zu Implementierungen von Wiederholungsstrategien und dazu, wie Sie sich für eine Strategie entscheiden können.

Alternativ können diese Optionen auch in Ihrem Code konfiguriert werden, wie in den folgenden Abschnitten gezeigt.

Geben Sie das Verhalten beim erneuten Versuch des Clients im Code an

Standardmäßig führt das AWS SDK for Ruby bis zu drei Wiederholungen mit 15 Sekunden zwischen den Wiederholungen durch, was insgesamt bis zu vier Versuchen entspricht. Eine Operation könnte in diesem Beispiel bis zum Timeout bis zu 60 Sekunden dauern.

Das folgende Beispiel erstellt einen Amazon S3 S3-Client in der Region und gibt an`us-west-2`, dass zwischen zwei Wiederholungen bei jedem Client-Vorgang fünf Sekunden gewartet werden sollen. Daher kann es Amazon S3 S3-Client-Vorgängen bis zu 15 Sekunden dauern, bis das Timeout eintritt.

```
 s3 = Aws::S3::Client.new(
   region: region,
   retry_limit: 2,
   retry_backoff: lambda { |c| sleep(5) }
)
```

Jede explizite Einstellung, die im Code oder auf einem Service-Client selbst festgelegt ist, hat Vorrang vor den Einstellungen in Umgebungsvariablen oder der gemeinsam genutzten `config` Datei.

Konfiguration von Observability-Funktionen im AWS SDK for Ruby

Beobachtbarkeit ist das Ausmaß, in dem der aktuelle Zustand eines Systems aus den von ihm ausgegebenen Daten abgeleitet werden kann. Die ausgegebenen Daten werden allgemein als Telemetrie bezeichnet. Das AWS SDK for Ruby kann die Traces als Telemetriesignal bereitstellen. Sie können eine Verbindung herstellen, um Telemetriedaten `TelemetryProvider` zu sammeln und an ein Observability-Backend zu senden. Das SDK unterstützt derzeit OpenTelemetry (OTel) als Telemetrieanbieter und OpenTelemetry bietet viele Möglichkeiten, Ihre Telemetriedaten zu exportieren, einschließlich der Verwendung von Amazon AWS X-Ray. CloudWatch Weitere Informationen zu OpenTelemetry Exportprogrammen für Ruby finden Sie unter Exporter auf der Website. OpenTelemetry

Standardmäßig zeichnet das SDK keine Telemetriedaten auf oder sendet sie aus. In diesem Thema wird erklärt, wie die Telemetrieausgabe konfiguriert und ausgegeben wird.

Telemetrie kann entweder für einen bestimmten Dienst oder global konfiguriert werden. Das SDK for Ruby stellt einen OpenTelemetry Anbieter bereit. Sie können auch einen benutzerdefinierten Telemetrieanbieter Ihrer Wahl definieren.

Konfiguration eines **`OTelProvider`** für einen Service-Client

Das SDK for Ruby bietet einen OpenTelemetry Anbieter namens `OTelProvider`. Im folgenden Beispiel wird der Telemetrieexport OpenTelemetry für den Amazon Simple Storage Service Service-Client konfiguriert. In diesem einfachen Beispiel OpenTelemetry wird die `OTEL_TRACES_EXPORTER` Umgebungsvariable from verwendet, um die Traces in die Konsolenausgabe zu exportieren, wenn Sie den Code ausführen. Weitere Informationen `OTEL_TRACES_EXPORTER` dazu finden Sie in der OpenTelemetry Dokumentation unter Exporter Selection.

```
require 'aws-sdk-s3'
require 'opentelemetry-sdk'
require 'opentelemetry-exporter-otlp'

ENV['OTEL_TRACES_EXPORTER'] ||= 'console'

OpenTelemetry::SDK.configure
```

```
otel_provider = Aws::Telemetry::OTelProvider.new
client = Aws::S3::Client.new(telemetry_provider: otel_provider)
client.list_buckets
```

Das vorherige Codebeispiel zeigt die Schritte zur Konfiguration der Trace-Ausgabe für einen Service-Client:

1. OpenTelemetry Abhängigkeiten erforderlich.
   a. `opentelemetry-sdk`zur Verwendung`Aws::Telemetry::OTelProvider`.
   b. `opentelemetry-exporter-otlp`zum Exportieren von Telemetriedaten.
2. Rufen Sie `OpenTelemetry::SDK.configure` auf, um das OpenTelemetry SDK mit seinen Standardkonfigurationen einzurichten.
3. Erstellen Sie mit dem SDK für den OpenTelemetry Anbieter von Ruby eine Instanz von, die `OTelProvider` als Konfigurationsoption an den Service-Client übergeben werden soll, den Sie verfolgen möchten.

   ```
   otel_provider = Aws::Telemetry::OTelProvider.new
   client = Aws::S3::Client.new(telemetry_provider: otel_provider)
   ```

Mithilfe dieser Schritte geben alle Methoden, die auf diesem Service-Client aufgerufen werden, Trace-Daten aus.

Ein Beispiel für die Trace-Ausgabe, die durch den Aufruf der Amazon S3 `list_buckets` S3-Methode generiert wurde, lautet wie folgt:

Beispiel für eine OpenTelemetry Trace-Ausgabe

```
#<struct OpenTelemetry::SDK::Trace::SpanData
 name="Handler.NetHttp",
 kind=:internal,
 status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
 parent_span_id="\xBFb\xC9\xFD\xA6F!\xE1",
 total_recorded_attributes=7,
 total_recorded_events=0,
 total_recorded_links=0,
 start_timestamp=1736190567061767000,
 end_timestamp=1736190567317160000,
 attributes=
  {"http.method"=>"GET",
```

```
    "net.protocol.name"=>"http",
    "net.protocol.version"=>"1.1",
    "net.peer.name"=>"s3.amazonaws.com",
    "net.peer.port"=>"443",
    "http.status_code"=>"200",
    "aws.request_id"=>"22HSH7NQTYMB5NHQ"},
  links=nil,
  events=nil,
  resource=
   #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
    @attributes=
     {"service.name"=>"unknown_service",
      "process.pid"=>37013,
      "process.command"=>"example.rb",
      "process.runtime.name"=>"ruby",
      "process.runtime.version"=>"3.3.0",
      "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
  [arm64-darwin23]",
      "telemetry.sdk.name"=>"opentelemetry",
      "telemetry.sdk.language"=>"ruby",
      "telemetry.sdk.version"=>"1.6.0"}>,
  instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
  name="aws.s3.client", version="">,
  span_id="\xEF%\x9C\xB5\x8C\x04\xDB\x7F",
  trace_id=" \xE7\xF1\xF8\x9D\e\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
  trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
  tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
#<struct OpenTelemetry::SDK::Trace::SpanData
  name="S3.ListBuckets",
  kind=:client,
  status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
  parent_span_id="\x00\x00\x00\x00\x00\x00\x00\x00",
  total_recorded_attributes=5,
  total_recorded_events=0,
  total_recorded_links=0,
  start_timestamp=1736190567054410000,
  end_timestamp=1736190567327916000,
  attributes={"rpc.system"=>"aws-api", "rpc.service"=>"S3", "rpc.method"=>"ListBuckets",
  "code.function"=>"list_buckets", "code.namespace"=>"Aws::Plugins::Telemetry"},
  links=nil,
  events=nil,
  resource=
   #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
    @attributes=
```

```
    {"service.name"=>"unknown_service",
     "process.pid"=>37013,
     "process.command"=>"example.rb",
     "process.runtime.name"=>"ruby",
     "process.runtime.version"=>"3.3.0",
     "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
[arm64-darwin23]",
     "telemetry.sdk.name"=>"opentelemetry",
     "telemetry.sdk.language"=>"ruby",
     "telemetry.sdk.version"=>"1.6.0"}>,
instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
name="aws.s3.client", version="">,
span_id="\xBFb\xC9\xFD\xA6F!\xE1",
trace_id=" \xE7\xF1\xF8\x9D\e\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
```

Die vorherige Trace-Ausgabe enthält zwei Datenbereiche. Jeder Trace-Eintrag enthält zusätzliche Metadaten über das Ereignis in einem oder mehreren Attributen.

Konfiguration eines **`OTelProvider`** für alle Service-Clients

Anstatt die Telemetrie für einen bestimmten Service-Client zu aktivieren, wie im vorherigen Abschnitt erklärt, haben Sie die Möglichkeit, die Telemetrie global zu aktivieren.

Um Telemetriedaten für alle AWS Service-Clients zu senden, können Sie den Telemetrieanbieter `Aws.config` vor dem Erstellen von Service-Clients auf dem Server einrichten.

```
otel_provider = Aws::Telemetry::OTelProvider.new
Aws.config[:telemetry_provider] = otel_provider
```

Bei dieser Konfiguration senden alle danach erstellten Service-Clients automatisch Telemetrie aus. Weitere Informationen zur Verwendung `Aws.config` zum Festlegen globaler Einstellungen finden Sie unter`Aws.config`.

Konfiguration eines benutzerdefinierten Telemetrieanbieters

Wenn Sie es nicht OpenTelemetry als Ihren Telemetrieanbieter verwenden möchten, unterstützt Sie das AWS SDK for Ruby bei der Implementierung eines benutzerdefinierten Anbieters. Es könnte hilfreich sein, die `OTelProvider`Implementierung, die im AWS SDK for Ruby GitHub

Ruby-Repository verfügbar ist, als Beispiel zu verwenden. Weitere Informationen finden Sie in den Hinweisen `Module: Aws::Telemetry`in der AWS SDK für Ruby API-Referenz.

Span-Attribute

Spuren sind das Ergebnis von Telemetrie. Spuren bestehen aus einer oder mehreren Bereichen. Spans haben Attribute, die zusätzliche Metadaten enthalten, die automatisch eingeschlossen werden, wenn sie für den Methodenaufruf geeignet sind. Im Folgenden finden Sie eine Liste der Attribute, die vom SDK for Ruby unterstützt werden, wobei:

- Attributname — der Name, der zur Bezeichnung der Daten verwendet wird, die im Trace erscheinen.
- Typ — der Datentyp des Werts.
- Beschreibung — eine Beschreibung dessen, wofür der Wert steht.

Attributname	Typ	Beschreibung
`error`	Boolesch	Stimmt, wenn die Arbeitsei nheit nicht erfolgreich war. Ansonsten „false“.
`exception.message`	Zeichenfolge	Die Ausnahme oder Fehlermel dung.
`exception.stacktrace`	Zeichenfolge	Ein Stacktrace, wie er von der Sprachlaufzeit bereitgestellt wird, falls verfügbar.
`exception.type`	Zeichenfolge	Der Typ (vollqualifizierter Name) der Ausnahme oder des Fehlers.
`rpc.system`	Zeichenfolge	Die Remotesystem-ID ist auf '`aws-api`' gesetzt.
`rpc.method`	Zeichenfolge	Der Name der Operation, die aufgerufen wird.

`rpc.service`	Zeichenfolge	Der Name des Remotedienstes.
`aws.request_id`	Zeichenfolge	Die in den Antwortheadern pro HTTP-Versuch zurückgegebene AWS Anforderungs-ID. Wenn möglich, wird die neueste Anforderungs-ID verwendet.
`code.function`	Zeichenfolge	Der Methoden- oder Funktionsname.
`code.namespace`	Zeichenfolge	Der Namespace, in dem definiert `code.function` ist.
`http.status_code`	Long	Der Statuscode der HTTP-Antwort.
`http.request_content_length`	Long	Die Größe des Payload-Hauptteils der Anfrage in Byte.
`http.response_content_length`	Long	Die Größe des Antwort-Payload-Hauptteils in Byte.
`http.method`	Zeichenfolge	Die HTTP-Anfragemethode.
`net.protocol.name`	Zeichenfolge	Der Name des Protokolls auf Anwendungsebene.
`net.protocol.version`	Zeichenfolge	Die Version des Anwendungsschichtprotokolls (z. B. 1.0, 1.1, 2.0).
`net.peer.name`	Zeichenfolge	Der logische Remote-Hostname.

`net.peer.port`	Zeichenfolge	Die logische Remote-Portnummer.

> **Tip**
>
> OpenTelemetry-Ruby verfügt über zusätzliche Implementierungen, die in das SDK für die bestehende Telemetrieunterstützung von Ruby integriert sind. Weitere Informationen finden Sie unter OpenTelemetry AWS-SDK Instrumentation im Repository. `open-telemetry` GitHub

Konfiguration von Einstellungen auf HTTP-Ebene im AWS SDK for Ruby

Einen nicht standardmäßigen Endpunkt einrichten

Die Region wird verwendet, um einen SSL-Endpunkt für AWS Anfragen zu erstellen. Wenn Sie in der ausgewählten Region einen nicht standardmäßigen Endpunkt verwenden müssen, fügen Sie einen `endpoint` Eintrag hinzu`Aws.config`. Sie können den auch festlegen, `endpoint:` wenn Sie einen Service-Client oder ein Ressourcenobjekt erstellen. Das folgende Beispiel erstellt ein Amazon S3 S3-Ressourcenobjekt auf dem `other_endpoint` Endpunkt.

```
s3 = Aws::S3::Resource.new(endpoint: other_endpoint)
```

Informationen zur Verwendung eines Endpunkts Ihrer Wahl für API-Anfragen und zur Beibehaltung dieser Auswahl finden Sie unter der Konfigurationsoption für dienstspezifische Endgeräte im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Das AWS SDK for Ruby verwenden

Dieser Abschnitt enthält Informationen zur Softwareentwicklung mit dem AWS SDK for Ruby, einschließlich der Verwendung einiger der erweiterten Funktionen des SDK.

Das Referenzhandbuch AWS SDKs und das Tools-Referenzhandbuch enthalten auch Einstellungen, Funktionen und andere grundlegende Konzepte, die AWS SDKs vielen von ihnen gemeinsam sind.

Themen

AWS-Service Anfragen mit dem AWS SDK for Ruby stellen

SDKs Verwenden Sie für den AWS-Services programmgesteuerten Zugriff jeweils eine Client-Klasse. AWS-Service Wenn Ihre Anwendung beispielsweise auf Amazon zugreifen muss EC2, erstellt Ihre Anwendung ein EC2 Amazon-Client-Objekt als Schnittstelle zu diesem Service. Anschließend verwenden Sie den Service-Client, um Anfragen an dieses zu stellen AWS-Service.

Um eine Anfrage an einen zu stellen AWS-Service, müssen Sie zunächst einen Service-Client erstellen und konfigurieren. Für jeden Code, den AWS-Service Sie verwenden, gibt es ein eigenes Gem und einen eigenen Typ, mit dem Sie interagieren können. Der Client stellt für jeden API-Vorgang, der vom Dienst verfügbar gemacht wird, eine Methode zur Verfügung.

Jeder Dienstclient benötigt einen AWS-Region und einen Anmeldeinformationsanbieter. Das SDK verwendet diese Werte, um Anfragen an die richtige Region für Ihre Ressourcen zu senden und Anfragen mit den richtigen Anmeldeinformationen zu signieren. Sie können diese Werte programmgesteuert im Code angeben oder sie automatisch aus der Umgebung laden lassen.

- Bei der Instanziierung einer Clientklasse müssen AWS Anmeldeinformationen angegeben werden. Informationen zur Reihenfolge, in der das SDK nach Authentifizierungsanbietern sucht, finden Sie unter. Kette von Anbietern von Anmeldeinformationen

- Das SDK hat eine Reihe von Orten (oder Quellen), die überprüft werden, um einen Wert für Konfigurationseinstellungen zu finden. Details hierzu finden Sie unter Vorrang der Einstellungen.

Das SDK for Ruby enthält Clientklassen, die Schnittstellen zu den bereitstellen AWS-Services. Jede Clientklasse unterstützt eine bestimmte Klasse AWS-Service und folgt der KonventionAws::*<service identifier>*::Client. `Aws::S3::Client`Stellt beispielsweise eine Schnittstelle zum Amazon Simple Storage Service Service und eine Schnittstelle zum Amazon Simple Queue Service Service `Aws::SQS::Client`bereit.

Alle Client-Klassen für alle AWS-Services sind Thread-sicher.

Sie können Konfigurationsoptionen direkt an Client- und Ressourcenkonstruktoren übergeben. Diese Optionen haben Vorrang vor der Umgebung und `Aws.config` den Standardeinstellungen.

```
# using a credentials object
ec2 = Aws::EC2::Client.new(region: 'us-west-2', credentials: credentials)
```

Verwenden des REPL-Dienstprogramms AWS SDK for Ruby

Das `aws-sdk` Gem enthält eine interaktive Read-Eval-Print-Loop (REPL) Befehlszeilenschnittstelle, über die Sie das SDK for Ruby testen und sofort die Ergebnisse sehen können. SDK for Ruby Gems sind unter RubyGems.org verfügbar.

Voraussetzungen

- Das AWS SDK for Ruby installieren.
- Das `aws-v3.rb`befindet sich im `aws-sdk-resources`Edelstein. Der `aws-sdk-resources` Edelstein ist auch im `aws-sdk`Haupt-Edelstein enthalten.
- Sie benötigen eine XML-Bibliothek, z. B. den `rexml` Edelstein.
- Obwohl das Programm mit der interaktiven Ruby-Shell (`irb`) funktioniert, empfehlen wir Ihnen, das `pry` Gem zu installieren, das eine leistungsfähigere REPL-Umgebung bietet.

Bundler-Setup

Wenn Sie Bundler verwenden, werden die folgenden Updates für `Gemfile` Sie die erforderlichen Gems beheben:

1. Öffnen Sie Ihre`Gemfile`, die Sie bei der Installation des AWS SDK for Ruby erstellt haben. Fügen Sie der Datei die folgenden Zeilen hinzu:

   ```
   gem "aws-sdk"
   gem "rexml"
   gem "pry"
   ```

2. Speichern Sie das Gemfile.
3. Installieren Sie die Abhängigkeiten, die in Ihrem`Gemfile`:

   ```
   $ bundle install
   ```

REPL ausführen

Sie können auf die REPL zugreifen, indem Sie sie von der Befehlszeile `aws-v3.rb` aus ausführen.

```
aws-v3.rb
```

Alternativ können Sie die HTTP-Verbindungsprotokollierung aktivieren, indem Sie das Verbose-Flag setzen. HTTP Wire Logging liefert Informationen über die Kommunikation zwischen dem AWS SDK for Ruby und AWS. Beachten Sie, dass das verbose-Flag auch zusätzlichen Aufwand verursacht, der dazu führen kann, dass Ihr Code langsamer ausgeführt wird.

```
aws-v3.rb -v
```

Das SDK for Ruby enthält Clientklassen, die Schnittstellen zu den bereitstellen AWS-Services. Jede Client-Klasse unterstützt eine bestimmte AWS-Service. In der REPL hat jede Serviceklasse einen Helfer, der ein neues Client-Objekt für die Interaktion mit diesem Dienst zurückgibt. Der Name des Helpers wird der Name des Dienstes sein, der in Kleinbuchstaben umgewandelt wurde. Die Namen der Amazon S3- und EC2 Amazon-Helper-Objekte lauten `ec2` beispielsweise `s3` und. Um die Amazon S3 S3-Buckets in Ihrem Konto aufzulisten, können Sie die Aufforderung `s3.list_buckets` eingeben.

Sie können `quit` in die REPL-Eingabeaufforderung tippen, um den Vorgang zu beenden.

Verwenden des AWS SDK for Ruby mit Ruby on Rails

Ruby on Rails bietet ein Webentwicklungs-Framework, das die Erstellung von Websites mit Ruby vereinfacht.

AWS bietet das `aws-sdk-rails` Juwel, um eine einfache Integration mit Rails zu ermöglichen. Sie können AWS Elastic Beanstalk, AWS OpsWorks AWS CodeDeploy, oder den AWS Rails Provisioner verwenden, um Ihre Rails-Anwendungen in der AWS Cloud bereitzustellen und auszuführen.

Informationen zur Installation und Verwendung des `aws-sdk-rails` Gems finden Sie im GitHub Repository https://github.com/aws/aws-sdk-rails.

Debuggen mithilfe von Wire-Trace-Informationen aus einem AWS SDK for Ruby Ruby-Client

Sie können Wire-Trace-Informationen von einem AWS Client abrufen, indem Sie den `http_wire_trace` booleschen Wert festlegen. Mithilfe von Wire-Trace-Informationen können Sie zwischen Kundenänderungen, Serviceproblemen und Benutzerfehlern unterscheiden. Wann`true`, die Einstellung zeigt an, was auf der Leitung gesendet wird. Im folgenden Beispiel wird ein Amazon S3 S3-Client erstellt, bei dem Wire Tracing zum Zeitpunkt der Client-Erstellung aktiviert ist.

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Mit dem folgenden Code und dem `bucket_name`-Argument zeigt die Ausgabe eine Nachricht an, die angibt, ob ein Bucket mit diesem Namen vorhanden ist.

```
require 'aws-sdk-s3'

s3 = Aws::S3::Resource.new(client: Aws::S3::Client.new(http_wire_trace: true))

if s3.bucket(ARGV[0]).exists?
  puts "Bucket #{ARGV[0]} exists"
else
  puts "Bucket #{ARGV[0]} does not exist"
end
```

Wenn der Bucket vorhanden ist, ähnelt die Ausgabe der folgenden. (Der HEAD-Zeile wurden zur besseren Lesbarkeit Zeilenumbrüche hinzugefügt.)

```
opening connection to bucket_name.s3-us-west-1.amazonaws.com:443...
opened
starting SSL for bucket_name.s3-us-west-1.amazonaws.com:443...
SSL established, protocol: TLSv1.2, cipher: ECDHE-RSA-AES128-GCM-SHA256
-> "HEAD / HTTP/1.1
    Accept-Encoding:
    User-Agent: aws-sdk-ruby3/3.171.0 ruby/3.2.2 x86_64-linux aws-sdk-s3/1.120.0
    Host: bucket_name.s3-us-west-1.amazonaws.com
    X-Amz-Date: 20230427T143146Z
/* omitted */
Accept: */*\r\n\r\n"
-> "HTTP/1.1 200 OK\r\n"
-> "x-amz-id-2: XxB2J+kpHgTjmMUwpkUI1EjaFSPxAjWRgkn/+z7YwWc/
iAX5E3OXRBzJ37cfc8T4D7ELC1KFELM=\r\n"
-> "x-amz-request-id: 5MD4APQQS815QVBR\r\n"
-> "Date: Thu, 27 Apr 2023 14:31:47 GMT\r\n"
-> "x-amz-bucket-region: us-east-1\r\n"
-> "x-amz-access-point-alias: false\r\n"
-> "Content-Type: application/xml\r\n"
-> "Server: AmazonS3\r\n"
-> "\r\n"
Conn keep-alive
Bucket bucket_name exists
```

Sie können Wiretracing auch nach der Client-Erstellung aktivieren.

```
s3 = Aws::S3::Client.new
s3.config.http_wire_trace = true
```

Weitere Informationen zu den Feldern in den gemeldeten Wire-Trace-Informationen finden Sie unter Erforderliche Anforderungsheader für Transfer Family.

Hinzufügen von Tests mit Stubbing zu Ihrer AWS SDK for Ruby Ruby-Anwendung

Erfahren Sie, wie Sie Client-Antworten und Client-Fehler in einer AWS SDK-Anwendung für Ruby abfragen.

Antworten von Kunden abfragen

Wenn Sie eine Antwort abfragen, deaktiviert das AWS SDK for Ruby den Netzwerkverkehr und der Client gibt Stubb-Daten (oder gefälschte) Daten zurück. Wenn Sie keine Stubb-Daten angeben, gibt der Client Folgendes zurück:

- Listen als leere Arrays
- Zuordnungen als leere Hashes
- Numerische Werte als Null
- Daten als now

Das folgende Beispiel gibt Stubbed-Namen für die Liste der Amazon S3 S3-Buckets zurück.

```
require 'aws-sdk'

s3 = Aws::S3::Client.new(stub_responses: true)

bucket_data = s3.stub_data(:list_buckets, :buckets => [{name:'aws-sdk'}, {name:'aws-
sdk2'}])
s3.stub_responses(:list_buckets, bucket_data)
bucket_names = s3.list_buckets.buckets.map(&:name)

# List each bucket by name
bucket_names.each do |name|
  puts name
end
```

Wenn Sie diesen Befehl ausführen, wird Folgendes angezeigt.

```
aws-sdk
aws-sdk2
```

> **Note**
>
> Nachdem Sie Stub-Daten bereitgestellt haben, gelten die Standardwerte nicht mehr für die verbleibenden Instance-Attribute. Das bedeutet, dass im vorherigen Beispiel das verbleibende Instance-`creation_date`-Attribut nicht now, sondern `nil` lautet.

Das AWS SDK for Ruby validiert Ihre Stubbed-Daten. Wenn Sie Daten vom falschen Typ übergeben, wird eine `ArgumentError`-Ausnahme ausgelöst. Angenommen, Sie haben z. B. anstelle der vorherigen Zuweisung zu `bucket_data` Folgendes verwendet:

```
bucket_data = s3.stub_data(:list_buckets, buckets:['aws-sdk', 'aws-sdk2'])
```

Das AWS SDK for Ruby löst zwei `ArgumentError` Ausnahmen aus.

```
expected params[:buckets][0] to be a hash
expected params[:buckets][1] to be a hash
```

Fehler beim Stubbing-Client

Sie können auch Fehler abfragen, die das AWS SDK for Ruby für bestimmte Methoden auslöst. Das folgende Beispiel zeigt `Caught Timeout::Error error calling head_bucket on aws-sdk` an.

```
require 'aws-sdk'

s3 = Aws::S3::Client.new(stub_responses: true)
s3.stub_responses(:head_bucket, Timeout::Error)

begin
  s3.head_bucket({bucket: 'aws-sdk'})
rescue Exception => ex
  puts "Caught #{ex.class} error calling 'head_bucket' on 'aws-sdk'"
end
```

Verwenden von paginierten Ergebnissen im AWS SDK for Ruby

Viele AWS Operationen geben gekürzte Ergebnisse zurück, wenn die Nutzlast zu groß ist, um sie in einer einzigen Antwort zurückzugeben. Stattdessen gibt der Dienst einen Teil der Daten und ein Token zurück, um den nächsten Satz von Elementen abzurufen. Dieses Muster wird als Paginierung bezeichnet.

Seitenbezogene Antworten sind aufzählbar

Die einfachste Methode zum Verarbeiten von Seiten mit Antwortdaten ist die Verwendung des integrierten Enumerators im Antwortobjekt, wie im folgenden Beispiel gezeigt.

```
s3 = Aws::S3::Client.new

s3.list_objects(bucket:'aws-sdk').each do |response|
  puts response.contents.map(&:key)
end
```

Dies ergibt ein Antwortobjekt pro ausgeführtem API-Aufruf und zählt Objekte in dem benannten Bucket auf. Das SDK ruft zusätzliche Seiten von Daten ab, um die Anforderung abzuschließen.

Manuelles Verarbeiten von Antworten auf Seiten

Wenn Sie das Blättern selbst verarbeiten möchten, verwenden Sie die `next_page?`-Methode, um zu bestätigen, dass mehr Seiten zum Abrufen vorhanden sind. Alternativ können Sie mit der `last_page?`-Methode überprüfen, ob keine weiteren Seiten zum Abrufen verfügbar sind.

Wenn es mehr Seiten gibt, verwenden Sie die `next_page`-Methode (`?` ist nicht vorhanden), um die nächste Ergebnisseite abzurufen, wie im folgenden Beispiel veranschaulicht.

```
s3 = Aws::S3::Client.new

# Get the first page of data
response = s3.list_objects(bucket:'aws-sdk')

# Get additional pages
while response.next_page? do
  response = response.next_page
  # Use the response data here...
end
```

> **Note**
>
> Wenn Sie die `next_page` Methode aufrufen und keine Seiten mehr abgerufen werden können, löst das SDK eine Aws:::PageableResponse: LastPageError Exception aus.

Klassen für ausgelagerte Daten

Seitenbezogene Daten im AWS SDK for Ruby werden von der PageableResponse Klasse Aws:: verarbeitet, die in Seahorse: :Client: :Response enthalten ist, um Zugriff auf ausgelagerte Daten zu ermöglichen.

Kellner im AWS SDK for Ruby verwenden

Waiter sind Hilfsprogrammmethoden, die einen bestimmten Status auf einem Client abfragen. Waiter-Objekte können fehlschlagen, nachdem eine Anzahl von Versuchen in einem für den Service-Client definierten Abrufintervall fehlgeschlagen ist. Ein Beispiel dafür, wie ein Kellner verwendet wird, finden Sie in der Methode create_table des Amazon DynamoDB Encryption Client im Code Examples Repository. AWS

Einen Kellner anrufen

Rufen Sie zum Aufrufen eines Waiter-Objekts `wait_until` auf einem Service-Client auf. Im folgenden Beispiel wartet ein Waiter-Objekt, bis die Instance `i-12345678` ausgeführt wird, bevor der Vorgang fortgesetzt wird.

```
ec2 = Aws::EC2::Client.new

begin
  ec2.wait_until(:instance_running, instance_ids:['i-12345678'])
  puts "instance running"
rescue Aws::Waiters::Errors::WaiterFailed => error
  puts "failed waiting for instance running: #{error.message}"
end
```

Der erste Parameter ist der Waiter-Name, der für den Service-Client spezifisch ist und die Operation angibt, auf die gewartet wird. Der zweite Parameter ist ein Hash von Parametern, die an die durch das Waiter-Objekt – welches gemäß dem Waiter-Namen variiert – aufgerufene Client-Methode übergeben werden.

Eine Liste der Operationen, auf die gewartet werden kann, und die jeweils aufgerufenen Client-Methoden finden Sie in der Dokumentation zu den `waiter_names`- und `wait_until`-Feldern für den von Ihnen verwendeten Client.

Fehlschläge warten

Waiter-Objekte können mit den folgenden Ausnahmen fehlschlagen.

Aws: :Waiters: :Fehler: FailureStateError

> Während des Wartens ist ein Fehlerstatus aufgetreten.

Aws: :Waiters: :Fehler: NoSuchWaiterError

Der angegebene Waiter-Name wurde für den verwendeten Client nicht definiert.

Aws: :Waiters: :Fehler: TooManyAttemptsError

Die Anzahl der Versuche hat den `max_attempts`-Wert des Waiter-Objekts überschritten.

Aws: :Waiters: :Fehler: UnexpectedError

Während des Wartens ist ein unerwarteter Fehler aufgetreten.

Aws: :Waiters: :Fehler: WaiterFailed

Während des Wartens wurde einer der Wartestatus überschritten oder es ist ein anderer Fehler aufgetreten.

Alle diese Fehler — außer — basieren `NoSuchWaiterError` auf. `WaiterFailed` Zum Abfangen von Fehlern in einem Waiter-Objekt verwenden Sie `WaiterFailed`, wie im folgenden Beispiel gezeigt.

```
rescue Aws::Waiters::Errors::WaiterFailed => error
  puts "failed waiting for instance running: #{error.message}"
end
```

Einen Kellner konfigurieren

Jedes Waiter-Objekt verfügt über ein Standard-Abrufintervall und eine maximale Anzahl von ausgeführten Versuchen, bevor die Steuerung an Ihr Programm zurückgegeben wird. Diese Werte legen Sie über die `max_attempts`- und `delay:`-Parameter in Ihrem `wait_until`-Aufruf fest. Der folgende Beispielcode wartet bis zu 25 Sekunden und fragt alle 5 Sekunden ab.

```
# Poll for ~25 seconds
client.wait_until(...) do |w|
  w.max_attempts = 5
  w.delay = 5
end
```

Zum Deaktivieren von Wartefehlern setzen Sie den Wert von einem der dieser beiden Parameter auf `nil`.

Einen Kellner verlängern

Wenn Sie das Verhalten von Waiter-Objekten ändern möchten, können Sie Callbacks registrieren, die vor jedem Abfrageversuch und vor dem Wartevorgang ausgelöst werden.

Das folgende Beispiel implementiert ein exponentielles Backoff in einem Waiter-Objekt durch die Verdopplung der Wartedauer bei jedem Versuch.

```
ec2 = Aws::EC2::Client.new

ec2.wait_until(:instance_running, instance_ids:['i-12345678']) do |w|
  w.interval = 0 # disable normal sleep
  w.before_wait do |n, resp|
    sleep(n ** 2)
  end
end
```

Im folgenden Beispiel wird die maximale Anzahl der Versuche deaktiviert. Stattdessen wird für 1 Stunde (3 600 Sekunden) gewartet, bevor der Vorgang fehlschlägt.

```
started_at = Time.now
client.wait_until(...) do |w|
  # Disable max attempts
  w.max_attempts = nil

  # Poll for one hour, instead of a number of attempts
  w.before_wait do |attempts, response|
    throw :failure if Time.now - started_at > 3600
  end
end
```

SDK for Ruby Ruby-Codebeispiele

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie AWS SDK für Ruby 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 Beispielkategorien, die zeigen, wie Bibliotheken oder dienstspezifische Funktionen genutzt werden können.

Dienstleistungen

- Aurora-Beispiele unter Verwendung von SDK für Ruby
- Auto-Scaling-Beispiele unter Verwendung von SDK für Ruby
- CloudTrail Beispiele mit SDK for Ruby
- CloudWatch Beispiele mit SDK for Ruby
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Ruby
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Ruby
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Ruby
- DynamoDB-Beispiele unter Verwendung von SDK für Ruby
- EC2 Amazon-Beispiele mit SDK for Ruby
- Beispiele für Elastic Beanstalk unter Verwendung von SDK für Ruby
- EventBridge Beispiele mit SDK for Ruby
- AWS Glue Beispiele mit SDK for Ruby
- IAM-Beispiele unter Verwendung von SDK für Ruby
- Kinesis-Beispiele unter Verwendung von SDK für Ruby
- AWS KMS Beispiele mit SDK for Ruby
- Lambda-Beispiele unter Verwendung von SDK für Ruby

- Beispiele für Amazon MSK unter Verwendung von SDK für Ruby
- Beispiele für Amazon Polly unter Verwendung von SDK für Ruby
- Beispiele für Amazon RDS unter Verwendung von SDK für Ruby
- Beispiele für Amazon S3 unter Verwendung von SDK für Ruby
- Beispiele für Amazon SES unter Verwendung von SDK für Ruby
- Beispiele für Amazon SES API v2 unter Verwendung von SDK für Ruby
- Beispiele für Amazon SNS unter Verwendung von SDK für Ruby
- Beispiele für Amazon SQS unter Verwendung von SDK für Ruby
- AWS STS Beispiele mit SDK for Ruby
- Beispiele für Amazon Textract unter Verwendung von SDKs für Ruby
- Codebeispiele für Amazon Translate unter Verwendung von SDKs für Ruby

Aurora-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby mit Aurora 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

- Erste Schritte

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Ruby

> **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 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Einzelheiten zur API finden Sie unter Describe DBClusters in der AWS SDK für Ruby API-Referenz.

Auto-Scaling-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby mit Auto Scaling 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

- Erste Schritte

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen können.

SDK für Ruby

> **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 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/#{group.max_size}/
#{group.desired_capacity}")
```

```
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für Ruby API-Referenz.

CloudTrail Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und allgemeine Szenarien implementieren CloudTrail.

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

`CreateTrail`

Das folgende Codebeispiel zeigt, wie Sie`CreateTrail`.

SDK für Ruby

> **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 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => "arn:aws:s3:::#{bucket_name}"
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:PutObject',
          'Resource' => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
          'Condition' => {
            'StringEquals' => {
```

```
            's3:x-amz-acl' => 'bucket-owner-full-control'
          }
        }
      }
    ]
  }.to_json

  s3_client.put_bucket_policy(
    bucket: bucket_name,
    policy: policy
  )
  puts "Successfully added policy to bucket #{bucket_name}"
end

begin
  cloudtrail_client.create_trail({
                                   name: trail_name, # required
                                   s3_bucket_name: bucket_name # required
                                 })

  puts "Successfully created trail: #{trail_name}."
rescue StandardError => e
  puts "Got error trying to create trail #{trail_name}:\n #{e}"
  puts e
  exit 1
end
```

- Einzelheiten zur API finden Sie CreateTrailin der AWS SDK für Ruby API-Referenz.

`DeleteTrail`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTrail`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: #{trail_name}"
rescue StandardError => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
end
```

- Einzelheiten zur API finden Sie DeleteTrailin der AWS SDK für Ruby API-Referenz.

`ListTrails`

Das folgende Codebeispiel zeigt die Verwendung`ListTrails`.

SDK für Ruby

> **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 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           #{trail.name}"
    puts "S3 bucket name: #{trail.s3_bucket_name}"
    puts
  end
```

- Einzelheiten zur API finden Sie ListTrailsin der AWS SDK für Ruby API-Referenz.

`LookupEvents`

Das folgende Codebeispiel zeigt die Verwendung`LookupEvents`.

SDK für Ruby

> **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 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts 'Resources:'

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ''
    end
  end
end
```

- Einzelheiten zur API finden Sie LookupEventsin der AWS SDK für Ruby API-Referenz.

CloudWatch Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und allgemeine Szenarien implementieren CloudWatch.

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

`DescribeAlarms`

Das folgende Codebeispiel zeigt, wie Sie`DescribeAlarms`.

SDK für Ruby

> **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 'aws-sdk-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
```

```
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der AWS SDK für Ruby API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für Ruby

> **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 cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts "Name:           #{alarm.alarm_name}"
      puts "State value:    #{alarm.state_value}"
      puts "State reason:   #{alarm.state_reason}"
      puts "Metric:         #{alarm.metric_name}"
      puts "Namespace:      #{alarm.namespace}"
      puts "Statistic:      #{alarm.statistic}"
      puts "Period:         #{alarm.period}"
      puts "Unit:           #{alarm.unit}"
      puts "Eval. periods:  #{alarm.evaluation_periods}"
      puts "Threshold:      #{alarm.threshold}"
```

```
      puts "Comp. operator: #{alarm.comparison_operator}"

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts 'Alarm actions:'
        alarm.alarm_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
         alarm.insufficient_data_actions.count.positive?
        puts 'Insufficient data actions:'
        alarm.insufficient_data_actions.each do |a|
          puts "  #{a}"
        end
      end

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: #{d.name}, Value: #{d.value}"
        end
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ''
```

```
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für Ruby API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
```

```
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
```

```
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für Ruby API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end

# Example usage:
def run_me
```

```
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
    'example.com',
    5_885.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für Ruby API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
```

```
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
```

```
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für Ruby API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für Ruby

> **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 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
```

```
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für Ruby

> **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 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
```

```
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Comprehend unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Comprehend Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

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

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

> Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:
>
> - Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
> - Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
> - Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
> - Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.
>
> Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.
>
> In diesem Beispiel verwendete Dienste
>
> - Amazon Comprehend
> - Lambda
> - Amazon Polly
> - Amazon Textract

- Amazon Translate

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby 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

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 Ruby

> **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 Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
```

```
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

DynamoDB-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby 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

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

- Serverless-Beispiele

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für Ruby

> **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 'aws-sdk-dynamodb'
require 'logger'

# DynamoDBManager is a class responsible for managing DynamoDB operations
# such as listing all tables in the current AWS account.
class DynamoDBManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all DynamoDB tables in the current AWS account.
  def list_tables
    @logger.info('Here are the DynamoDB tables in your account:')

    paginator = @client.list_tables(limit: 10)
    table_names = []

    paginator.each_page do |page|
      page.table_names.each do |table_name|
        @logger.info("- #{table_name}")
        table_names << table_name
      end
    end
```

```
    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  dynamodb_client = Aws::DynamoDB::Client.new
  manager = DynamoDBManager.new(dynamodb_client)
  manager.list_tables
end
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für Ruby API-Referenz.

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 Ruby

> **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 Klasse, die eine DynamoDB-Tabelle enthält.

```
  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
  # Gets sample movie data, either from a local file or by first downloading it from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip'
```

```
      )
      movie_json = ''
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
  table_name = "doc-example-table-movies-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  dynamodb_wrapper = DynamoDBBasics.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Add a new record to the DynamoDB table.')
  my_movie = {}
  my_movie[:title] = CLI::UI::Prompt.ask('Enter the title of a movie to add to the
table. E.g. The Matrix')
  my_movie[:year] = CLI::UI::Prompt.ask('What year was it released? E.g. 1989').to_i
  my_movie[:rating] = CLI::UI::Prompt.ask('On a scale of 1 - 10, how do you rate it?
E.g. 7').to_i
  my_movie[:plot] = CLI::UI::Prompt.ask('Enter a brief summary of the plot. E.g. A
man awakens to a new reality.')
  dynamodb_wrapper.add_item(my_movie)
```

```
  puts("\nNew record added:")
  puts JSON.pretty_generate(my_movie).green
  print "Done!\n".green

  new_step(3, 'Update a record in the DynamoDB table.')
  my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
  response = dynamodb_wrapper.update_item(my_movie)
  puts("Updated '#{my_movie[:title]}' with new attributes:")
  puts JSON.pretty_generate(response).green
  print "Done!\n".green

  new_step(4, 'Get a record from the DynamoDB table.')
  puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
  response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
  puts JSON.pretty_generate(response).green
  print "Done!\n".green

  new_step(5, 'Write a batch of items into the DynamoDB table.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(5, 'Query for a batch of items by key.')
  loop do
    release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
    results = dynamodb_wrapper.query_items(release_year)
    if results.any?
      puts("There were #{results.length} movies released in #{release_year}:")
      results.each do |movie|
        print "\t #{movie['title']}".green
      end
      break
    else
      continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
      break unless continue.eql?('y')
    end
  end
```

```
  print "\nDone!\n".green

  new_step(6, 'Scan for a batch of items using a filter expression.')
  years = {}
  years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
 2018:')
  years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
  releases = dynamodb_wrapper.scan_items(years)
  if !releases.empty?
    puts("Found #{releases.length} movies.")
    count = Question.ask(
      'How many do you want to see? ', method(:is_int), in_range(1, releases.length)
    )
    puts("Here are your #{count} movies:")
    releases.take(count).each do |release|
      puts("\t#{release['title']}")
    end
  else
    puts("I don't know about any movies released between #{years[:start]} "\
         "and #{years[:end]}.")
  end
  print "\nDone!\n".green

  new_step(7, 'Delete an item from the DynamoDB table.')
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
")
  if answer.eql?('y')
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, 'Delete the DynamoDB table.')
  answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
  if answer.eql?('y')
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
```

```
  true
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -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 Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lesen Sie einen Stapel von Elementen mithilfe von PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
```

```
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

Löschen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
```

```
  end
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Ruby API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must contain
 at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
      movies[index, slice_size].each do |movie|
```

```
        movie_items.append({ put_request: { item: movie } })
      end
      @dynamo_resource.client.batch_write_item({ request_items: { @table.name =>
 movie_items } })
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:"
    )
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für Ruby API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end
```

```
  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für Ruby API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
```

```
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für Ruby API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für Ruby API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
```

```
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für Ruby API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wählen Sie ein einzelnes Element mithilfe von PartiQL aus.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table = @dynamodb.table(table_name)
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Aktualisieren Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Fügen Sie ein einzelnes Element mithilfe von PartiQL hinzu.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, { 'plot': plot, 'rating': rating }]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Löschen Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
```

```
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Ruby API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
```

```
    @table.get_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für Ruby API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Finden Sie heraus, ob eine Tabelle vorhanden ist.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
```

```
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für Ruby API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
```

```
      item: {
        'year' => movie[:year],
        'title' => movie[:title],
        'info' => { 'plot' => movie[:plot], 'rating' => movie[:rating] }
      }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für Ruby API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
```

```
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: { ':year' => year }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für Ruby -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt, wie man es benutzt`Scan`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
```

```
    movies = []
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für Ruby -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt, wie man es benutzt`UpdateItem`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
```

```
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)
    response = @table.update_item(
      key: { 'year' => movie[:year], 'title' => movie[:title] },
      update_expression: 'set info.rating=:r',
      expression_attribute_values: { ':r' => movie[:rating] },
      return_values: 'UPDATED_NEW'
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
#{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.attributes
  end
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für Ruby API-Referenz.

Szenarien

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 Ruby

> **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 eine Tabelle erstellt wird und PartiQL-Stapelabfragen ausgeführt werden.

```
  table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  sdk = DynamoDBPartiQLBatch.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Populate DynamoDB table with movie data.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a batch of items from the movies table.')
  puts "Let's select some popular movies for side-by-side comparison."
  response = sdk.batch_execute_select([['Mean Girls', 2004], ['Goodfellas', 1977],
['The Prancing of the Lambs', 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, 'Delete a batch of items from the movies table.')
  sdk.batch_execute_write([['Mean Girls', 2004], ['Goodfellas', 1977], ['The
Prancing of the Lambs', 2005]])
  print "\nDone!\n".green
```

```
  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Ruby 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 Ruby

> **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, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLSingle.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
```

```
  end

  new_step(2, 'Populate DynamoDB table with movie data.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a single item from the movies table.')
  response = sdk.select_item_by_title('Star Wars')
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print response.items.first.to_s.yellow
  print "\n\nDone!\n".green

  new_step(4, 'Update a single item from the movies table.')
  puts "Let's correct the rating on The Big Lebowski to 10.0."
  sdk.update_rating_by_title('The Big Lebowski', 1998, 10.0)
  print "\nDone!\n".green

  new_step(5, 'Delete a single item from the movies table.')
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title('The Silence of the Lambs', 1991)
  print "\nDone!\n".green

  new_step(6, 'Insert a new item into the movies table.')
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item('The Prancing of the Lambs', 2005, 'A movie about happy
 livestock.', 5.0)
  print "\nDone!\n".green

  new_step(7, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Ruby 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 Ruby

> **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 Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

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 Ruby

> **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 Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

EC2 Amazon-Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Ruby mit Amazon verwenden 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

Erste Schritte

Hallo Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon beginnen können EC2.

SDK für Ruby

> **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 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
```

```
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
    instances = []

    paginator.each_page do |page|
      page.reservations.each do |reservation|
        reservation.instances.each do |instance|
          instances << instance
        end
      end
    end

    instances
  end

  # Prints details of the given EC2 instances.
  #
  # @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
 print.
  def print_instances(instances)
    instances.each do |instance|
      @logger.info("Instance ID: #{instance.instance_id}")
      @logger.info("Instance Type: #{instance.instance_type}")
      @logger.info("Public IP: #{instance.public_ip_address}")
      @logger.info("Public DNS Name: #{instance.public_dns_name}")
      @logger.info("\n")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
```

```
  manager.list_instances
end
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für Ruby API-Referenz.

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für Ruby API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
```

```
  'Error'
end
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für Ruby API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
```

```
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
```

```
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
```

```
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'

  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für Ruby API-Referenz.

`CreateRouteTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteTable`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
```

```
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
```

```
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts 'Route table created and associated.'
  else
    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateRouteTablein der AWS SDK für Ruby API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
```

```
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
```

```
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
 perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
 perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
  print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

  print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

  print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
 perm.ip_ranges.count.positive?
  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
```

```
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:        #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?
```

```
  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http, to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
 description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
 from_port_http, to_port_http, cidr_ip_range_http)
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh, from_port_ssh,
 to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
 security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
```

```
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
  security_group_id = create_security_group(ec2_client, group_name, description,
 vpc_id)
  puts 'Could not create security group. Skipping this step.' if security_group_id
 == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
 to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
 ip_protocol, from_port, to_port,
                                               cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
end

def display_help
  puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
    'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
    'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
```

```
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp', '80',
 '80',
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für Ruby API-Referenz.

`CreateSubnet`

Das folgende Codebeispiel zeigt die Verwendung`CreateSubnet`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
```

```
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
```

```
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-west-2a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
```

```
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateSubnetin der AWS SDK für Ruby API-Referenz.

`CreateVpc`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpc`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
```

```
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
```

```
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateVpcin der AWS SDK für Ruby API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für Ruby API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
```

```
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end
```

```
# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
 '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS SDK für Ruby API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für Ruby API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end
```

```
# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_started?(ec2_client, instance_id)

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für Ruby API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end
```

```
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für Ruby API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Ruby

> **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 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
     response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
```

```
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für Ruby API-Referenz.

Beispiele für Elastic Beanstalk unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Elastic Beanstalk Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

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

`DescribeApplications`

Das folgende Codebeispiel zeigt, wie Sie. `DescribeApplications`

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
```

```
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Error listing environments for application #{application_name}:
 #{e.message}")
  end
end
```

- Einzelheiten zur API finden Sie DescribeApplicationsin der AWS SDK für Ruby API-Referenz.

`ListAvailableSolutionStacks`

Das folgende Codebeispiel zeigt die Verwendung`ListAvailableSolutionStacks`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
  end

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
```

```
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Einzelheiten zur API finden Sie ListAvailableSolutionStacksin der AWS SDK für Ruby API-Referenz.

`UpdateApplication`

Das folgende Codebeispiel zeigt die Verwendung`UpdateApplication`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end
```

```
  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key: key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end

  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
```

```
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie UpdateApplicationin der AWS SDK für Ruby API-Referenz.

EventBridge Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und allgemeine Szenarien implementieren EventBridge.

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

Erstellen und Auslösen einer Regel

Das folgende Codebeispiel zeigt, wie eine Regel in Amazon erstellt und ausgelöst wird EventBridge.

SDK für Ruby

> **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 die Funktionen in der richtigen Reihenfolge auf.

```
require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'
```

Überprüfen Sie, ob das angegebene Amazon Simple Notification Service (Amazon SNS)-Thema unter den für diese Funktion bereitgestellten Themen vorhanden ist.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
```

```
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Überprüfen Sie, ob das angegebene Thema unter den für den Aufrufer in Amazon SNS verfügbaren Themen vorhanden ist.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
```

```
    while response.next_page?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Erstellen Sie ein Thema in Amazon SNS und abonnieren Sie dann eine E-Mail-Adresse, um Benachrichtigungen zu diesem Thema zu erhalten.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
```

```
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Prüfen Sie, ob die angegebene AWS Identity and Access Management (IAM-) Rolle unter den Rollen existiert, die für diese Funktion bereitgestellt wurden.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  false
end
```

Überprüfen Sie, ob die angegebene Rolle unter den für den Aufrufer in IAM verfügbaren Rollen vorhanden ist.

```
# Checks whether the specified role exists among those available to the
```

```
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

      if role_found?(response.roles, role_arn)
        puts 'Role found.'
        return true
      end
    end
  end
  puts 'Role not found.'
  false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end
```

Erstellen Sie eine Rolle in IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
```

```
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
```

```
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  'Error'
end
```

Überprüft, ob die angegebene EventBridge Regel unter den für diese Funktion bereitgestellten Regeln existiert.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  false
end
```

Überprüft, ob die angegebene Regel zu den Regeln gehört, die dem Anrufer zur Verfügung stehen. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

      if rule_found?(response.rules, rule_name)
        puts 'Rule found.'
        return true
      end
    end
  end
  puts 'Rule not found.'
  false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  false
end
```

Erstellen Sie eine Regel in EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
```

```
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
```

```
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
     put_targets_response.failed_entry_count.positive?
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end
```

Prüfen Sie, ob die angegebene Protokollgruppe zu den Protokollgruppen gehört, die dem Anrufer in Amazon CloudWatch Logs zur Verfügung stehen.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

Erstellen Sie eine Protokollgruppe in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
```

```
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  false
end
```

Schreiben Sie ein Ereignis in einen Protokollstream in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
```

```
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

Starten Sie eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance neu und fügen Sie Informationen über die zugehörige Aktivität zu einem Protokollstream in CloudWatch Logs hinzu.

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
```

```
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
```

```
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end
```

Zeigt Informationen zur Aktivität für eine Regel in an EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
```

```
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
    unit: 'Count'
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Zeigt Protokollinformationen für alle Protokolldatenströme in einer Protokollgruppe „ CloudWatch Protokolle“ an.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
```

```
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end
```

Zeigt dem Anrufer eine Erinnerung an, alle zugehörigen AWS Ressourcen, die er nicht mehr benötigt, manuell zu bereinigen.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - PutEvents
  - PutRule

AWS Glue Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby 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

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für Ruby

> **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 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
```

```
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für Ruby API-Referenz.

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.
- Führen 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 Ruby

> **Note**
>
> Es gibt mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die die im Szenario verwendeten AWS Glue Funktionen umschließt.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
 crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
```

```
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

```
  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
```

```
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

```
 # Deletes a job with the specified name.
 #
 # @param job_name [String] The name of the job to delete.
 # @return [void]
 def delete_job(job_name)
   @glue_client.delete_job(job_name: job_name)
 rescue Aws::Glue::Errors::ServiceError => e
   @logger.error("Glue could not delete job: \n#{e.message}")
 end

 # Deletes a table with the specified name.
 #
 # @param database_name [String] The name of the catalog database in which the
table resides.
 # @param table_name [String] The name of the table to be deleted.
 # @return [void]
 def delete_table(database_name, table_name)
   @glue_client.delete_table(database_name: database_name, name: table_name)
 rescue Aws::Glue::Errors::ServiceError => e
   @logger.error("Glue could not delete job: \n#{e.message}")
 end

 # Removes a specified database from a Data Catalog.
 #
 # @param database_name [String] The name of the database to delete.
 # @return [void]
 def delete_database(database_name)
   @glue_client.delete_database(name: database_name)
 rescue Aws::Glue::Errors::ServiceError => e
   @logger.error("Glue could not delete database: \n#{e.message}")
 end

 # Uploads a job script file to an S3 bucket.
 #
 # @param file_path [String] The local path of the job script file.
 # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
 # @return [void]
 def upload_job_script(file_path, bucket_resource)
   File.open(file_path) do |file|
     bucket_resource.client.put_object({
                                         body: file,
                                         bucket: bucket_resource.name,
```

```
                                        key: file_path
                                      })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end

  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
```

```
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
 t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]['name'],
 @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end

def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'
```

```
  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-east-1'),
 iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

Erstellen Sie ein ETL-Skript, das AWS Glue zum Extrahieren, Transformieren und Laden von Daten während Jobausführungen verwendet wird.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
```

```
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
```

```
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung von`CreateCrawler`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
 crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
```

```
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für Ruby API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
```

```
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für Ruby API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für Ruby API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
```

```
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für Ruby API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für Ruby API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
```

```
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für Ruby API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
```

```
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für Ruby API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
 if not found.
```

```
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für Ruby API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für Ruby API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die VerwendungGetJobRuns.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
```

```
  end
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für Ruby API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für Ruby API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für Ruby API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für Ruby API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für Ruby API-Referenz.

IAM-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby 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

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt, wie Sie mit IAM beginnen.

SDK für Ruby

> **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 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Ruby API-Referenz.

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 Ruby

> **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 IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client
```

```
# @param [Aws::IAM::Client] iam_client: The AWS IAM client.
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
end

# Waits for the specified number of seconds.
#
# @param duration [Integer] The number of seconds to wait.
def wait(duration)
  puts('Give AWS time to propagate resources...')
  sleep(duration)
end

# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_client.create_user(user_name: user_name).user
  @logger.info("Created demo user named #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info('Tried and failed to create demo user.')
  @logger.info("\t#{e.code}: #{e.message}")
  @logger.info("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
  @logger.info("Created accesskey pair for user #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create access keys for user #{user.user_name}.")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  user_key
```

```
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
```

```
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
```

```
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated into
a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
```

```
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are deleted
  # before the user is deleted.
```

```
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
 user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
```

```
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -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 Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für Ruby API-Referenz.

`AttachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachUserPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
```

```
  end
```

- Einzelheiten zur API finden Sie AttachUserPolicyin der AWS SDK für Ruby API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
```

```
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
```

```
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für Ruby API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases
```

```
    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für Ruby API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für Ruby API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an error
occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn
```

```
    policy_arns.each do |policy_arn|
      @iam_client.attach_role_policy(
        role_name: role_name,
        policy_arn: policy_arn
      )
    end

    role_arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating role: #{e.message}")
    nil
  end
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für Ruby API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates a service-linked role
  #
  # @param service_name [String] The service name to create the role for.
  # @param description [String] The description of the service-linked role.
  # @param suffix [String] Suffix for customizing role name.
  # @return [String] The name of the created role
  def create_service_linked_role(service_name, description, suffix)
    response = @iam_client.create_service_linked_role(
      aws_service_name: service_name, description: description, custom_suffix:
suffix
    )
    role_name = response.role.role_name
    @logger.info("Created service-linked role #{role_name}.")
```

```
    role_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für Ruby API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates a user and their login profile
#
# @param user_name [String] The name of the user
# @param initial_password [String] The initial password for the user
# @return [String, nil] The ID of the user if created, or nil if an error occurred
def create_user(user_name, initial_password)
  response = @iam_client.create_user(user_name: user_name)
  @iam_client.wait_until(:user_exists, user_name: user_name)
  @iam_client.create_login_profile(
    user_name: user_name,
    password: initial_password,
    password_reset_required: true
  )
  @logger.info("User '#{user_name}' created successfully.")
  response.user.user_id
rescue Aws::IAM::Errors::EntityAlreadyExists
  @logger.error("Error creating user '#{user_name}': user already exists.")
```

```
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für Ruby API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
```

```
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
```

```
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für Ruby API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end
```

```
  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für Ruby API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes a role and its attached policies.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  # Detach and delete attached policies
  @iam_client.list_attached_role_policies(role_name: role_name).each do |response|
    response.attached_policies.each do |policy|
      @iam_client.detach_role_policy({
                                       role_name: role_name,
                                       policy_arn: policy.policy_arn
                                     })
      # Check if the policy is a customer managed policy (not AWS managed)
      unless policy.policy_arn.include?('aws:policy/')
        @iam_client.delete_policy({ policy_arn: policy.policy_arn })
        @logger.info("Deleted customer managed policy #{policy.policy_name}.")
      end
    end
  end

  # Delete the role
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Deleted role #{role_name}.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS SDK für Ruby API-Referenz.

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
```

```
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für Ruby API-Referenz.

`DeleteServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServiceLinkedRole`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes a service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id
    )
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)

    sleep(3)
  end
end

# Handles deletion error
```

```
  #
  # @param e [Aws::Errors::ServiceError] The error encountered during deletion
  # @param role_name [String] The name of the role attempted to delete
  def handle_deletion_error(e, role_name)
    return if e.code == 'NoSuchEntity'

    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der AWS SDK für Ruby API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a user and their associated resources
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für Ruby API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a user and their associated resources
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der AWS SDK für Ruby API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
```

```
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

```
  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für Ruby API-Referenz.

`DetachUserPolicy`

Das folgende Codebeispiel zeigt die VerwendungDetachUserPolicy.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Detaches a policy from a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The ARN of the policy to detach
  # @return [Boolean] true if the policy was successfully detached, false otherwise
  def detach_user_policy(user_name, policy_arn)
    @iam_client.detach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
```

```
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
    false
  end
```

- Einzelheiten zur API finden Sie DetachUserPolicyin der AWS SDK für Ruby API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is: #{response.password_policy.to_h}")
```

```
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für Ruby API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
```

```
end
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für Ruby API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die VerwendungGetRole.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                                role_name: name
                              }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für Ruby API-Referenz.

`GetUser`

Das folgende Codebeispiel zeigt die VerwendungGetUser.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- Einzelheiten zur API finden Sie GetUserin der AWS SDK für Ruby API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end
```

```
  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für Ruby API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
```

```
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für Ruby API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
```

```
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
```

```
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für Ruby API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für Ruby API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
```

```
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

```
  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Ruby API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
```

```
    []
  end
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für Ruby API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count

      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für Ruby API-Referenz.

`ListSAMLProviders`

Das folgende Codebeispiel zeigt die Verwendung`ListSAMLProviders`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie unter Liste SAMLProviders in der AWS SDK für Ruby API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
```

```
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für Ruby API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für Ruby API-Referenz.

`PutUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutUserPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
                                user_name: username,
                                policy_name: policy_name,
                                policy_document: policy_document
                              })
  @logger.info("Policy #{policy_name} created for user #{username}.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  false
end
```

- Einzelheiten zur API finden Sie PutUserPolicyin der AWS SDK für Ruby API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateServerCertificate`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end
```

```
    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für Ruby API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to '#{new_name}':
#{e.message}")
  false
end
```

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für Ruby API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby 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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
```

```
  return data
end
```

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
```

```
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

AWS KMS Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und allgemeine Szenarien implementieren AWS KMS.

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

`CreateKey`

Das folgende Codebeispiel zeigt, wie Sie`CreateKey`.

SDK für Ruby

> **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 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für Ruby API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Ruby

> **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 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK für Ruby -API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt, wie man es benutzt`Encrypt`.

SDK für Ruby

> **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 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK für Ruby -API-Referenz.

`ReEncrypt`

Das folgende Codebeispiel zeigt, wie man es benutzt`ReEncrypt`.

SDK für Ruby

> **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 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.
```

```
blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
sourceCiphertextBlob = [blob].pack('H*')

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Einzelheiten zur API finden Sie ReEncryptin der AWS SDK für Ruby API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby 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

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen können.

SDK für Ruby

> **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 'aws-sdk-lambda'

# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
  Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
 necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end
```

```
  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Ruby API-Referenz.

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 Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die erforderlichen IAM-Berechtigungen für eine Lambda-Funktion ein, die Protokolle schreiben kann.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
  role_policy = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Principal': { 'Service': 'lambda.amazonaws.com' },
        'Action': 'sts:AssumeRole'
      }
    ]
  }
  role = @iam_client.create_role(
    role_name: iam_role_name,
```

```
      assume_role_policy_document: role_policy.to_json
    )
    @iam_client.attach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    wait_for_role_to_exist(iam_role_name)
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    sleep(10)
    [role, role_policy.to_json]
  end

  def destroy_iam_role(iam_role_name)
    @iam_client.detach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    @iam_client.delete_role(role_name: iam_role_name)
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  end

  def wait_for_role_to_exist(iam_role_name)
    @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  end
```

Definieren Sie einen Lambda-Handler, der eine als Aufrufparameter bereitgestellte Zahl inkrementiert.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
```

```
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
                   Logger::DEBUG
                 when 'info'
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Komprimieren (ZIP) Sie Ihre Lambda-Funktion in ein Bereitstellungspaket.

```
  # Creates a Lambda deployment package in .zip format.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda file
 and zip.
  # @return: The deployment package.
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?('lambda_function.zip')
      File.delete('lambda_function.zip')
      @logger.debug('Deleting old zip: lambda_function.zip')
    end
    Zip::File.open('lambda_function.zip', create: true) do |zipfile|
      zipfile.add('lambda_function.rb', "#{source_file}.rb")
    end
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read('lambda_function.zip').to_s
```

```
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
  end
```

Erstellen Sie eine neue Lambda-Funktion.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Rufen Sie Ihre Lambda-Funktion mit optionalen Laufzeitparametern auf.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

Aktualisieren Sie die Konfiguration Ihrer Lambda-Funktion, um eine neue Umgebungsvariable einzufügen.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       'LOG_LEVEL' => log_level
                                                     }
                                                   }
                                                 })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Aktualisieren Sie den Code Ihrer Lambda-Funktion mit einem anderen Bereitstellungspaket, das anderen Code enthält.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

Listen Sie alle vorhandenen Lambda-Funktionen mithilfe des eingebauten Paginators auf.

```
  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
```

```
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

Löschen Sie eine bestimmte Lambda-Funktion.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateFunction`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
 code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
```

```
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für Ruby API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
```

```
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print 'Done!'.green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für Ruby API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
```

```
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für Ruby API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
```

```
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für Ruby -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt, wie man es benutzt`ListFunctions`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Ruby API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
```

```
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für Ruby API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
```

```
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für Ruby API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.

- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

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 Ruby

> **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 Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
```

```
endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
port = ENV['Port']           # 3306
user = ENV['DBUser']
region = ENV['DBRegion']     # 'us-east-1'
db_name = ENV['DBName']

credentials = Aws::Credentials.new(
  ENV['AWS_ACCESS_KEY_ID'],
  ENV['AWS_SECRET_ACCESS_KEY'],
  ENV['AWS_SESSION_TOKEN']
)
rds_client = Aws::RDS::AuthTokenGenerator.new(
  region: region,
  credentials: credentials
)

token = rds_client.auth_token(
  endpoint: endpoint+ ':' + port,
  user_name: user,
  region: region
)

begin
  conn = Mysql2::Client.new(
    host: endpoint,
    username: user,
    password: token,
    port: port,
    database: db_name,
    sslca: '/var/task/global-bundle.pem',
    sslverify: true,
    enable_cleartext_plugin: true
  )
  a = 3
  b = 2
  result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
  puts result
  conn.close
  {
    statusCode: 200,
    body: result.to_json
  }
rescue => e
  puts "Database connection failed due to #{e}"
```

```
  end
end
```

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
```

```
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

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 Ruby

> **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 Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

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 Ruby

> **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 Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

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 Ruby

> **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 Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

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 Ruby

> **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 Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
```

```
end
```

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 Ruby

> **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 Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

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 Ruby

> **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 SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Beispiele für Amazon MSK unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

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 Ruby

> **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 Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
```

```
      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Beispiele für Amazon Polly unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon Polly verwenden. AWS SDK für Ruby

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

`DescribeVoices`

Das folgende Codebeispiel zeigt, wie Sie`DescribeVoices`.

SDK für Ruby

> **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 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie DescribeVoicesin der AWS SDK für Ruby API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für Ruby

> **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 'aws-sdk-polly' # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie ListLexiconsin der AWS SDK für Ruby API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für Ruby

> **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 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
```

```
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                   output_format: 'mp3',
                                   text: contents,
                                   voice_id: 'Joanna'
                                 })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin der AWS SDK für Ruby API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon RDS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon RDS verwenden. AWS SDK für Ruby

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

Hello Amazon RDS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon RDS.

SDK für Ruby

> **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 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
```

```
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Einzelheiten zur API finden Sie unter Describe DBInstances in der AWS SDK für Ruby API-Referenz.

Aktionen

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für Ruby

> **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 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Einzelheiten zur API finden Sie unter DBSnapshot In der AWS SDK für Ruby API-Referenz erstellen.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Ruby

> **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 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Einzelheiten zur API finden Sie unter Describe DBInstances in der AWS SDK für Ruby API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Ruby

> **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 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
```

```
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Einzelheiten zur API finden Sie unter DBParameterGruppen beschreiben in der AWS SDK für Ruby API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für Ruby

> **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 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
```

```
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Einzelheiten zur API finden Sie unter Describe DBParameters in der AWS SDK für Ruby API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für Ruby

> **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 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
```

```
end
```

- Einzelheiten zur API finden Sie unter Describe DBSnapshots in der AWS SDK für Ruby API-Referenz.

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 Ruby

> **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 Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
```

```
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Beispiele für Amazon S3 unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon S3 verwenden. AWS SDK für Ruby

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

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen können.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Ruby 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 Ruby

> **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 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
```

```
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
```

```
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
```

```
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
 __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt, wie man es benutzt`CopyObject`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
```

```
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Kopieren Sie ein Objekt und fügen Sie dem Zielobjekt eine serverseitige Verschlüsselung hinzu.

```
require 'aws-sdk-s3'
```

```
# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object
```

```
  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für Ruby API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
```

```
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für Ruby API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für Ruby API-Referenz.

`DeleteBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketCors`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin der AWS SDK für Ruby API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK für Ruby API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
```

```
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für Ruby API-Referenz.

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketCors`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
```

```
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- Einzelheiten zur API finden Sie GetBucketCorsin der AWS SDK für Ruby API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
```

```
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK für Ruby API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Ruby

> **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.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
```

```
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rufen Sie ein Objekt ab und melden Sie seinen serverseitigen Verschlüsselungsstatus.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
```

```
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für Ruby API-Referenz.

`HeadObject`

Das folgende Codebeispiel zeigt die Verwendung`HeadObject`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
```

```
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie HeadObjectin der AWS SDK für Ruby API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end
```

```
run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Ruby API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
```

```
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für Ruby API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
```

```
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie PutBucketCorsin der AWS SDK für Ruby API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK für Ruby API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für Ruby

> **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 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
```

```
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für Ruby API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Ruby

> **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 eine Datei mit einem verwalteten Uploader (Object.upload_file) hoch.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei mithilfe von Object.put hoch.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei mithilfe von Object.put hoch und fügen Sie eine serverseitige Verschlüsselung hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
```

```
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für Ruby 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 Ruby

> **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 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
```

```
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
 #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
 'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

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 Ruby

> **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 Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Beispiele für Amazon SES unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

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

`GetIdentityVerificationAttributes`

Das folgende Codebeispiel zeigt, wie Sie`GetIdentityVerificationAttributes`.

SDK für Ruby

> **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 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                               identity_type: 'EmailAddress'
                             })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })
```

```
  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin der AWS SDK für Ruby API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für Ruby

> **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 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                               identity_type: 'EmailAddress'
                             })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status
```

```
  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für Ruby API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Ruby

> **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 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</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-ruby/">'\
```

```
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"
```

```
# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für Ruby API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für Ruby

> **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 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
```

```
end
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon SES API v2 unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Ruby mit Amazon SES API v2 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.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendEmail`

Das folgende Codebeispiel zeigt, wie Sie`SendEmail`.

SDK für Ruby

> **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 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.
```

```
# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
      destination: {
        to_addresses: [recipient_email]
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon SNS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

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

`CreateTopic`

Das folgende Codebeispiel zeigt, wie Sie`CreateTopic`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to create an Amazon Simple Notification Service (SNS)
 topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
```

```
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Ruby API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
 Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end
```

```
  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für Ruby API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Ruby

> **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 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Ruby API-Referenz.

`Publish`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
```

```
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für Ruby -API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt, wie man es benutzt`SetTopicAttributes`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
```

```
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"
```

```
  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für Ruby API-Referenz.

`Subscribe`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Ruby

> **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 'aws-sdk-sns'
require 'logger'

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
```

```
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info('Subscription created successfully.')
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = 'email'
  endpoint = 'EMAIL_ADDRESS' # Should be replaced with a real email address
  topic_arn = 'TOPIC_ARN'    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info('Creating the subscription.')
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error('Subscription creation failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für Ruby -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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Beispiele für Amazon SQS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby mit Amazon SQS 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.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt, wie Sie`ChangeMessageVisibility`.

SDK für Ruby

> **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 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages: 10
                                                      })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
```

```
                                    visibility_timeout: 30 # This message will not
 be visible for 30 seconds after first receipt.
                                  })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                       queue_url: queue_url,
                                                       max_number_of_messages: 10
                                                     })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für Ruby API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require 'aws-sdk-sqs'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
```

```
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für Ruby API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Ruby

> **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 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für Ruby API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Ruby

> **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 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
```

```
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
```

```
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für Ruby API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Ruby

> **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 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
```

```
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end
```

```
# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für Ruby API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Ruby

> **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 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
```

```
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für Ruby API-Referenz.

`SendMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`SendMessageBatch`.

SDK für Ruby

> **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 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
```

```
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- Einzelheiten zur API finden Sie SendMessageBatchin der AWS SDK für Ruby API-Referenz.

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 Ruby

> **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 Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

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 Ruby

> **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 SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS Beispiele mit SDK for Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und allgemeine Szenarien implementieren AWS STS.

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

`AssumeRole`

Das folgende Codebeispiel zeigt, wie Sie`AssumeRole`.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
```

```
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Textract unter Verwendung von SDKs für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Textract Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

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 Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHubProjekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Codebeispiele für Amazon Translate unter Verwendung von SDKs für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Translate Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

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

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract

- Amazon Translate

Migration von AWS SDK for Ruby Version 1 oder 2 zu AWS SDK for Ruby Version 3

Dieses Thema enthält Informationen, die Ihnen bei der Migration von Version 1 oder 2 des AWS SDK for Ruby auf Version 3 helfen sollen.

Side-by-side Verwendung

Es ist nicht notwendig, die Version 1 oder 2 des AWS SDK for Ruby durch Version 3 zu ersetzen. Sie können sie in derselben Anwendung gemeinsam verwenden. Weitere Informationen dazu finden Sie in diesem Blog-Beitrag.

Ein kurzes Beispiel.

```
require 'aws-sdk-v1' # version 1
require 'aws-sdk'    # version 2
require 'aws-sdk-s3' # version 3

s3 = AWS::S3::Client.new # version 1
s3 = Aws::S3::Client.new # version 2 or 3
```

Sie müssen vorhandenen, funktionierenden Code der Version 1 oder 2 nicht umschreiben, um die SDK-Version 3 zu verwenden. Es ist eine gültige Strategie für die Migration, nur neuen Code für die SDK-Version 3 zu schreiben.

Allgemeine Unterschiede

Version 3 unterscheidet sich von Version 2 in einem wesentlichen Aspekt.

- Jeder Service ist als separates Gem verfügbar.

Version 2 unterscheidet sich von Version 1 in mehreren wichtigen Punkten.

- Anderer Root-Namespace — `Aws` versus`AWS`. Dies ermöglicht die side-by-side Verwendung.
- `Aws.config` – Jetzt ein Standard-Ruby-Hash anstelle von einer Methode.

- Strikte Konstruktoroptionen – Beim Erstellen von einem Client- oder Ressourcenobjekt in der SDK-Version 1 werden unbekannte Konstruktoroptionen ignoriert. In Version 2 lösen unbekannte Konstruktoroptionen einen `ArgumentError` aus. Beispiel:

```
# version 1
AWS::S3::Client.new(http_reed_timeout: 10)
# oops, typo'd option is ignored

# version 2
Aws::S3::Client.new(http_reed_timeout: 10)
# => raises ArgumentError
```

Unterschiede zwischen den Kunden

Es gibt keine Unterschiede zwischen den Clientklassen in Version 2 und Version 3.

Zwischen Version 1 und Version 2 haben die Client-Klassen die wenigsten externen Unterschiede. Viele Service-Clients haben nach der Clientkonstruktion kompatible Schnittstellen. Einige wichtige Unterschiede:

- `Aws::S3::Client`- Die Amazon S3 S3-Clientklasse der Version 1 wurde von Hand codiert. Version 2 wird aus einem Service-Modell generiert. Die Methodennamen und Eingaben in Version 2 unterscheiden sich deutlich.
- `Aws::EC2::Client` – Version 2 verwendet Pluralnamen für Ausgabelisten. Version 1 verwendet das `_set`-Suffix. Beispiel:

```
# version 1
resp = AWS::EC2::Client.new.describe_security_groups
resp.security_group_set
#=> [...]

# version 2
resp = Aws::EC2::Client.new.describe_security_groups
resp.security_groups
#=> [...]
```

- `Aws::SWF::Client` – Version 2 verwendet strukturierte Antworten, wo Version 1 Standard-Ruby-Hashes nutzt.

- Umbenennungen von Service-Klassen – Version 2 verwendet für mehrere Services einen anderen Namen:
  - `AWS::SimpleWorkflow` ist jetzt `Aws::SWF`.
  - `AWS::ELB` ist jetzt `Aws::ElasticLoadBalancing`.
  - `AWS::SimpleEmailService` ist jetzt `Aws::SES`.
- Client-Konfigurationsoptionen — Einige der Konfigurationsoptionen von Version 1 wurden in Version 2 umbenannt. Andere werden entfernt oder ersetzt. Hier sind die wichtigsten Änderungen:
  - `:use_ssl` wurde entfernt. Version 2 verwendet überall SSL. Zum Deaktivieren von SSL müssen Sie einen `:endpoint` konfigurieren, der `http://` verwendet.
  - `:ssl_ca_file` ist jetzt `:ssl_ca_bundle`
  - `:ssl_ca_path` ist jetzt `:ssl_ca_directory`
  - `:ssl_ca_store` hinzugefügt.
  - `:endpoint` muss jetzt ein vollqualifizierter HTTP- oder HTTPS-URI anstelle eines Hostnamens sein.
  - `:*_port`-Optionen für die einzelnen Services wurden entfernt und jetzt durch `:endpoint` ersetzt.
  - `:user_agent_prefix` ist jetzt `:user_agent_suffix`

Unterschiede bei den Ressourcen

Es gibt keine Unterschiede zwischen den Ressourcenschnittstellen in Version 2 und Version 3.

Es gibt signifikante Unterschiede zwischen den Ressourcenschnittstellen in Version 1 und Version 2. Version 1 war vollständig handcodiert. Die Ressourcenschnittstellen in Version 2 werden hingegen aus einem Modell generiert. Die Ressourcenschnittstellen in Version 2 sind wesentlich konsistenter. Einige der systemischen Unterschiede sind:

- Separate Ressourcenklasse — In Version 2 ist der Dienstname ein Modul, keine Klasse. In diesem Modul ist es die Ressourcenschnittstelle:

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- Verweise auf Ressourcen – Die SDK-Version 2 trennt Sammlungen und einzelne Ressourcen-Getter in zwei verschiedene Methoden:

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- Batch-Operationen — In Version 1 waren alle Batch-Operationen handcodierte Dienstprogramme. In Version 2 sind viele Stapeloperationen automatisch generierte Stapelverarbeitungsoperationen über die API. Die Stapelverarbeitungsschnittstellen in Version 2 unterscheiden sich stark von Version 1.

Sicherheit für AWS SDK for Ruby

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, was AWS-Service Sie verwenden, und anderen Faktoren wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Themen

Datenschutz im AWS SDK for Ruby

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 der Konsole, der API oder auf andere AWS-Services Weise arbeiten oder diese 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.

Identity and Access Management für AWS SDK for Ruby

AWS Identity and Access Management (IAM) ist ein Service von Amazon Web Services (AWS), der einem Administrator hilft, den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (Berechtigungen besitzt) ist, um AWS-Services Ressourcen zu nutzen. IAM ist ein Dienst AWS-Service , den Sie ohne zusätzliche Kosten nutzen können.

Um AWS das SDK für den Zugriff auf Ruby zu verwenden AWS, benötigen Sie ein AWS Konto und AWS Anmeldeinformationen. Um die Sicherheit Ihres AWS -Kontos zu erhöhen, empfehlen wir, dass Sie einen IAM-Benutzer verwenden, um Anmeldeinformationen bereitzustellen, statt Ihre AWS -Konto-Anmeldeinformationen zu verwenden.

Einzelheiten zur Arbeit mit IAM finden Sie unter IAM.

Eine Übersicht über die IAM-Benutzer und Informationen dazu, warum sie für die Sicherheit Ihres Kontos wichtig sind, finden Sie unter AWS -Sicherheitsanmeldeinformationen in der Allgemeinen Referenz zu Amazon Web Services.

AWS Das SDK for Ruby folgt dem Modell der gemeinsamen Verantwortung durch die spezifischen Amazon Web Services (AWS) -Dienste, die es unterstützt. AWS-Service Sicherheitsinformationen finden Sie auf der Seite mit der AWS-Service Sicherheitsdokumentation. AWS-Services Diese Informationen sind im Rahmen der AWS Compliance-Bemühungen des Compliance-Programms aufgeführt.

Konformitätsprüfung für AWS SDK for Ruby

AWS Das SDK for Ruby folgt dem Modell der gemeinsamen Verantwortung durch die spezifischen Amazon Web Services (AWS) -Dienste, die es unterstützt. AWS-Service Sicherheitsinformationen finden Sie auf der Seite mit der AWS-Service Sicherheitsdokumentation. AWS-Services Diese Informationen sind im Rahmen der AWS Compliance-Bemühungen des Compliance-Programms aufgeführt.

Die Sicherheit und Konformität der Dienste von Amazon Web Services (AWS) wird von externen Prüfern im Rahmen mehrerer AWS Compliance-Programme bewertet. Dazu gehören SOC, PCI, FedRAMP, HIPAA und andere. AWS bietet unter AWS Services in Scope by Compliance Program eine häufig aktualisierte Liste der AWS-Services im Umfang der jeweiligen Compliance-Programme aufgeführten Programme.

Prüfberichte von Drittanbietern stehen Ihnen zum Herunterladen zur Verfügung unter AWS Artifact. Weitere Informationen finden Sie unter Berichte in AWS Artifact herunterladen.

Weitere Informationen zu AWS Compliance-Programmen finden Sie unter AWS Compliance-Programme.

Ihre Compliance-Verantwortung bei der Verwendung von AWS SDK for Ruby für den Zugriff auf hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS-Service Wenn Ihre Verwendung von an AWS-Service die Einhaltung von Standards wie HIPAA, PCI oder FedRAMP unterliegt, AWS bietet Ihnen Ressourcen, die Ihnen helfen:

- Schnellstartanleitungen zu Sicherheit und Compliance — Implementierungsleitfäden, in denen architektonische Überlegungen erörtert und Schritte zur Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen beschrieben werden. AWS
- Referenz für berechtigte HIPAA-Services – Listet berechtigte HIPAA-Services auf. Nicht alle sind HIPAA-fähig AWS-Services .
- AWS Ressourcen zur Einhaltung von Vorschriften — Eine Sammlung von Arbeitsmappen und Leitfäden, die möglicherweise auf Ihre Branche und Ihren Standort zutreffen.
- AWS Config — Ein Service, der bewertet, wie gut Ihre Ressourcenkonfigurationen internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub — Ein umfassender Überblick über Ihren Sicherheitsstatus innerhalb AWS , anhand dessen Sie überprüfen können, ob Sie die Sicherheitsstandards und Best Practices der Sicherheitsbranche einhalten.

Resilienz für AWS SDK for Ruby

Die globale Infrastruktur von Amazon Web Services (AWS) basiert auf AWS-Regionen Availability Zones.

AWS-Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, 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 Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

AWS Das SDK for Ruby folgt dem Modell der gemeinsamen Verantwortung durch die spezifischen Amazon Web Services (AWS) -Dienste, die es unterstützt. AWS-Service Sicherheitsinformationen finden Sie auf der Seite mit der AWS-Service Sicherheitsdokumentation. AWS-Services Diese Informationen sind im Rahmen der AWS Compliance-Bemühungen des Compliance-Programms aufgeführt.

Infrastruktursicherheit für AWS SDK for Ruby

AWS Das SDK for Ruby folgt dem Modell der gemeinsamen Verantwortung durch die spezifischen Amazon Web Services (AWS) -Dienste, die es unterstützt. AWS-Service Sicherheitsinformationen finden Sie auf der Seite mit der AWS-Service Sicherheitsdokumentation. AWS-Services Diese Informationen sind im Rahmen der AWS Compliance-Bemühungen des Compliance-Programms aufgeführt.

Informationen zu AWS Sicherheitsprozessen finden Sie im Whitepaper AWS: Überblick über Sicherheitsprozesse.

Erzwingen einer TLS-Mindestversion im AWS SDK for Ruby

Die Kommunikation zwischen dem AWS SDK for Ruby und AWS ist mit Secure Sockets Layer (SSL) oder Transport Layer Security (TLS) gesichert. Alle Versionen von SSL und Versionen von TLS vor 1.2 weisen Sicherheitslücken auf, die die Sicherheit Ihrer Kommunikation mit gefährden können AWS. Aus diesem Grund sollten Sie sicherstellen, dass Sie das AWS SDK for Ruby mit einer Version von Ruby verwenden, die TLS Version 1.2 oder höher unterstützt.

Ruby verwendet die OpenSSL-Bibliothek, um HTTP-Verbindungen zu sichern. Unterstützte Versionen von Ruby (1.9.3 und höher), die über Systempaketmanager (`yum`, und andere)`apt`, ein offizielles Installationsprogramm oder Ruby-Manager (rbenv, RVM und andere) installiert werden, enthalten in der Regel OpenSSL 1.0.1 oder höher, das TLS 1.2 unterstützt.

Bei Verwendung mit einer unterstützten Version von Ruby mit OpenSSL 1.0.1 oder höher bevorzugt das AWS SDK for Ruby TLS 1.2 und verwendet die neueste Version von SSL oder TLS, die sowohl vom Client als auch vom Server unterstützt wird. Dies ist immer mindestens TLS 1.2 für. AWS-Services(Das SDK verwendet die Ruby-Klasse `Net::HTTP` mit `use_ssl=true`.)

Überprüfung der OpenSSL-Version

Um sicherzustellen, dass Ihre Ruby-Installation OpenSSL 1.0.1 oder höher verwendet, geben Sie den folgenden Befehl ein.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Eine alternative Möglichkeit, die OpenSSL-Version zu erhalten, besteht darin, die ausführbare Datei `openssl` direkt abzufragen. Suchen Sie zuerst mit dem folgenden Befehl die entsprechende ausführbare Datei.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

Die Ausgabe sollte als Speicherort der OpenSSL-Installation `--with-openssl-dir=/path/to/openssl` angeben. Notieren Sie sich diesen Pfad. Um die Version von OpenSSL zu überprüfen, geben Sie die folgenden Befehle ein.

```
cd /path/to/openssl
bin/openssl version
```

Diese letztere Methode funktioniert möglicherweise nicht mit allen Installationen von Ruby.

Aktualisierung der TLS-Unterstützung

Wenn die von Ihrer Ruby-Installation verwendete Version von OpenSSL älter als 1.0.1 ist, aktualisieren Sie Ihre Ruby- oder OpenSSL-Installation mit Ihrem Systempaketmanager, Ruby-Installer oder Ruby-Manager, wie in der Installationsanleitung von Ruby beschrieben. Wenn Sie Ruby aus dem Quellcode installieren, installieren Sie zuerst das neueste OpenSSL und übergeben Sie es dann, `--with-openssl-dir=/path/to/upgraded/openssl` wenn es ausgeführt wird `./configure`.

Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2)

> **Note**
>
> Wenn Sie Version 2 des S3-Verschlüsselungsclients verwenden und zu Version 3 migrieren möchten, finden Sie weitere Informationen unterMigration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3).

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 des AWS SDK for Ruby in Ihrer 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 zu lesen

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 jeder Hauptversion des AWS SDK for Ruby.

AWS SDK for Ruby Version 3 aktualisieren

Version 3 ist die neueste Version des AWS SDK für Ruby. Um diese Migration abzuschließen, müssen Sie Version 1.76.0 oder höher des `aws-sdk-s3` Gems verwenden.

Installation über die Befehlszeile

Verwenden Sie bei Projekten, die das `aws-sdk-s3` Gem installieren, die Versionsoption, um zu überprüfen, ob die Mindestversion 1.76.0 installiert ist.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Gemfiles verwenden

Für Projekte, die ein Gemfile zur Verwaltung von Abhängigkeiten verwenden, legen Sie die Mindestversion des `aws-sdk-s3` Gems auf 1.76.0 fest. Beispiel:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Ändern Sie Ihr Gemfile.
2. Führen Sie `bundle update aws-sdk-s3`. Führen `bundle info aws-sdk-s3` Sie den Befehl aus, um Ihre Version zu überprüfen.

AWS SDK for Ruby Version 2 aktualisieren

Version 2 des AWS SDK for Ruby wird am 21. November 2021 in den Wartungsmodus wechseln. Um diese Migration abzuschließen, müssen Sie Version 2.11.562 oder höher des aws-sdk-Gems verwenden.

Installation über die Befehlszeile

Verwenden Sie bei Projekten, die das `aws-sdk` Gem über die Befehlszeile installieren, die Versionsoption, um zu überprüfen, ob die Mindestversion von 2.11.562 installiert ist.

```
gem install aws-sdk -v '>= 2.11.562'
```

Gemfiles verwenden

Für Projekte, die ein Gemfile zur Verwaltung von Abhängigkeiten verwenden, legen Sie die Mindestversion des `aws-sdk` Gems auf 2.11.562 fest. Beispiel:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Ändern Sie Ihr Gemfile. Wenn Sie eine Gemfile.lock-Datei haben, löschen oder aktualisieren Sie sie.
2. Führen Sie `bundle update aws-sdk`. Führen Sie den Befehl aus, um Ihre Version zu überprüfen. `bundle info aws-sdk`

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.

Bevor Sie Ihren Code für die Verwendung des V2-Verschlüsselungsclients aktualisieren, stellen Sie sicher, dass Sie die vorherigen Schritte befolgt haben und die `aws-sdk-s3` Gem-Version 2.11.562 oder höher verwenden.

> **Note**
>
> Lesen Sie beim Entschlüsseln mit AES-GCM das gesamte Objekt bis zum Ende, bevor Sie die entschlüsselten Daten verwenden. Dadurch wird überprüft, ob das Objekt seit der Verschlüsselung nicht geändert wurde.

Konfiguration von V2-Verschlüsselungsclients

Der EncryptionV2: :Client erfordert eine zusätzliche Konfiguration. Ausführliche Informationen zur Konfiguration finden Sie in der EncryptionV2: :Client-Dokumentation oder in den Beispielen weiter unten in diesem Thema.

1. Die Key-Wrap-Methode und der Inhaltsverschlüsselungsalgorithmus müssen bei der Client-Erstellung angegeben werden. Wenn Sie ein neues erstellen`EncryptionV2::Client`, müssen Sie Werte für `key_wrap_schema` und angeben`content_encryption_schema`.

`key_wrap_schema`- Wenn Sie verwenden AWS KMS, muss dies auf eingestellt sein`:kms_context`. Wenn Sie einen symmetrischen Schlüssel (AES) verwenden, muss dieser auf `:aes_gcm` eingestellt sein. Wenn Sie einen asymmetrischen Schlüssel (RSA) verwenden, muss dieser auf eingestellt sein. `:rsa_oaep_sha1`

`content_encryption_schema`- Dieser Wert muss auf:aes_gcm_no_padding gesetzt sein.

2. security_profile muss bei der Client-Erstellung angegeben werden. Wenn Sie ein neues erstellen`EncryptionV2::Client`, müssen Sie einen Wert für angeben. `security_profile` Der Parameter security_profile bestimmt die Unterstützung für das Lesen von Objekten, die mit der älteren Version V1 geschrieben wurden. `Encryption::Client` Es gibt zwei Werte ::v2 und:v2_and_legacy. Um die Migration zu unterstützen, setzen Sie den Wert to:v2_and_legacy. *`security_profile`* Verwenden Sie:v2 nur für die Entwicklung neuer Anwendungen.

3. AWS KMS key ID wird standardmäßig erzwungen. In Version 1 wurde die `kms_key_id` zur Erstellung des Clients verwendete Datei nicht an die AWS KMS `Decrypt call` weitergegeben.

`Encryption::Client` AWS KMS kann diese Informationen aus Metadaten abrufen und sie dem symmetrischen Chiffretext-Blob hinzufügen. In V2, e`NCryptionV2: :Client`, wird die kms_key_id an den Decrypt-Aufruf übergeben, und der Aufruf schlägt fehl, wenn sie nicht mit dem Schlüssel übereinstimmt, der zum AWS KMS Verschlüsseln des Objekts verwendet wurde. Wenn Ihr Code zuvor darauf beruhte, dass kein bestimmtes festgelegt wurde, dann entweder bei der Client-Erstellung oder bei Aufrufen. `kms_key_id` `kms_key_id: :kms_allow_decrypt_with_any_cmk` `kms_allow_decrypt_with_any_cmk: true` `get_object`

Beispiel: Verwendung eines symmetrischen Schlüssels (AES)

Vor der Migration

```
client = Aws::S3::Encryption::Client.new(encryption_key: aes_key)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Nach der Migration

```
client = Aws::S3::EncryptionV2::Client.new(
  encryption_key: rsa_key,
  key_wrap_schema: :rsa_oaep_sha1, # the key_wrap_schema must be rsa_oaep_sha1 for
 asymmetric keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
 )
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No changes
```

Beispiel: Verwendung AWS KMS mit kms_key_id

Vor der Migration

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Nach der Migration

```
client = Aws::S3::EncryptionV2::Client.new(
```

```
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No change
```

Beispiel: Verwendung AWS KMS ohne kms_key_id

Vor der Migration

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Nach der Migration

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:
 true) # To allow decrypting with any cmk
```

Alternative nach der Migration

Wenn Sie Objekte nur mit dem S2-Verschlüsselungsclient lesen und entschlüsseln (niemals schreiben und verschlüsseln), verwenden Sie diesen Code.

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: :kms_allow_decrypt_with_any_cmk, # set kms_key_id to allow all get_object
 requests to use any cmk
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
```

```
)
resp = client.get_object(bucket: bucket, key: key) # No change
```

Migration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3)

> **Note**
>
> Wenn Sie Version 1 des S3-Verschlüsselungsclients verwenden, müssen Sie zuerst zu V2 migrieren, bevor Sie zu V3 migrieren. Anweisungen Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2) zur Migration von V1 zu V2 finden Sie unter.

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. V3 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

Version 3 des Amazon S3 S3-Verschlüsselungsclients führt AES GCM mit Key Commitment für mehr Sicherheit ein. Dieser neue Verschlüsselungsalgorithmus bietet Schutz vor Manipulation von Datenschlüsseln und gewährleistet die Integrität verschlüsselter Daten. Die Migration zu V3 erfordert eine sorgfältige Planung, um die Anwendungsverfügbarkeit und den Datenzugriff während des gesamten Prozesses aufrechtzuerhalten.

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Stellen Sie zunächst eine aktualisierte Version des AWS SDK for Ruby in Ihrer 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. Dazu gehören die Konfiguration von Commitment-Richtlinien und die Aktualisierung Ihres Codes, um die neuen Client-Konfigurationsoptionen nutzen zu können.

Wenn Sie noch nicht von V1 auf V2 migriert haben, müssen Sie diese Migration zuerst abschließen. Ausführliche Anweisungen Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2) zur Migration von V1 zu V2 finden Sie unter.

Die Funktionen von V3 verstehen

Version 3 des Amazon S3 S3-Verschlüsselungsclients führt zwei wichtige Sicherheitsfunktionen ein: Commitment Policies und AES GCM with Key Commitment. Das Verständnis dieser Funktionen ist für die Planung Ihrer Migrationsstrategie und die Gewährleistung der Sicherheit Ihrer verschlüsselten Daten unerlässlich.

Richtlinien für Verpflichtungen

Commitment-Richtlinien steuern, wie der Verschlüsselungsclient bei Verschlüsselungs- und Entschlüsselungsvorgängen mit der Schlüsselzusage umgeht. Key Commitment stellt sicher, dass verschlüsselte Daten nur mit genau dem Schlüssel entschlüsselt werden können, mit dem sie verschlüsselt wurden, und schützt so vor bestimmten Arten von kryptografischen Angriffen.

Der V3-Verschlüsselungsclient unterstützt drei Commitment Policy-Optionen:

`FORBID_ENCRYPT_ALLOW_DECRYPT`

Diese Richtlinie verschlüsselt Objekte ohne Schlüsselzusage und ermöglicht die Entschlüsselung sowohl von Objekten mit als auch ohne Schlüsselzusage.

- Verschlüsselungsverhalten: Objekte werden ohne Schlüsselzuweisung verschlüsselt, wobei dieselbe Algorithmussuite wie V2 verwendet wird.
- Entschlüsselungsverhalten: Kann Objekte entschlüsseln, die mit oder ohne Schlüsselbindung verschlüsselt wurden.
- Auswirkungen auf die Sicherheit: Diese Richtlinie erzwingt keine Schlüsselbindung und ermöglicht möglicherweise Manipulationen. Objekte, die mit dieser Richtlinie verschlüsselt wurden, profitieren nicht von den erweiterten Sicherheitsvorkehrungen von Key Commitment. Verwenden Sie diese Richtlinie nur während der Migration, wenn Sie die Kompatibilität mit dem V2-Verschlüsselungsverhalten aufrechterhalten müssen.
- Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können von allen V2- und V3-Implementierungen des S3-Verschlüsselungsclients gelesen werden.

`REQUIRE_ENCRYPT_ALLOW_DECRYPT`

Diese Richtlinie verschlüsselt Objekte mit Schlüsselzusage und ermöglicht die Entschlüsselung sowohl von Objekten mit als auch ohne Schlüsselzusage.

- Verschlüsselungsverhalten: Objekte werden mit Key Commitment unter Verwendung von AES GCM mit Key Commitment verschlüsselt.
- Entschlüsselungsverhalten: Kann Objekte entschlüsseln, die mit oder ohne Key Commitment verschlüsselt wurden, wodurch Abwärtskompatibilität gewährleistet ist.
- Auswirkungen auf die Sicherheit: Neue Objekte profitieren vom Key-Commitment-Schutz, während bestehende Objekte ohne Key Commitment weiterhin gelesen werden können. Dadurch wird ein Gleichgewicht zwischen Sicherheit und Abwärtskompatibilität während der Migration gewährleistet.
- Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können nur von der V3- und den neuesten V2-Implementierungen des S3-Verschlüsselungsclients gelesen werden.

`REQUIRE_ENCRYPT_REQUIRE_DECRYPT`

Diese Richtlinie verschlüsselt Objekte mit Key Commitment und erlaubt nur die Entschlüsselung von Objekten, die mit Key Commitment verschlüsselt wurden.

- Verschlüsselungsverhalten: Objekte werden mit Key Commitment unter Verwendung von AES GCM mit Key Commitment verschlüsselt.
- Entschlüsselungsverhalten: Es können nur Objekte entschlüsselt werden, die mit Key Commitment verschlüsselt wurden. Versuche, Objekte ohne Schlüsselzuweisung zu entschlüsseln, schlagen fehl.
- Auswirkungen auf die Sicherheit: Diese Richtlinie bietet ein Höchstmaß an Sicherheit, da bei allen Vorgängen die Schlüsselbindung durchgesetzt wird. Verwenden Sie diese Richtlinie erst, nachdem alle Objekte mit Key Commitment erneut verschlüsselt wurden und alle Clients auf Version 3 aktualisiert wurden.
- Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können nur von der V3- und den neuesten V2-Implementierungen des S3-Verschlüsselungsclients gelesen werden. Diese Richtlinie verhindert auch das Lesen von Objekten, die von V2- oder V1-Clients verschlüsselt wurden.

> **Note**
>
> Beginnen Sie bei der Planung Ihrer Migration damit, die Abwärtskompatibilität aufrechtzuerhalten und gleichzeitig die Sicherheitsvorteile `REQUIRE_ENCRYPT_ALLOW_DECRYPT` zu nutzen, die sich aus dem Einsatz

von Key Commitment für neue Objekte ergeben. Wechseln Sie erst zu, `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` nachdem alle Objekte erneut verschlüsselt und alle Clients auf Version 3 aktualisiert wurden.

AES GCM mit zentraler Verpflichtung

AES GCM with Key Commitment (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) ist ein neuer Verschlüsselungsalgorithmus, der in Version 3 eingeführt wurde und mehr Sicherheit bietet, indem er vor Manipulation von Datenschlüsseln schützt. Für die Planung Ihrer Migration ist es wichtig zu verstehen, wie dieser Algorithmus funktioniert und wann er angewendet wird.

Wie **`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`** unterscheidet es sich von früheren Algorithmen

Frühere Versionen des S3-Verschlüsselungsclients verwendeten AES CBC oder AES GCM ohne Schlüsselbindung, um den Datenschlüssel in den Befehlsdateien zu verschlüsseln. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`fügt dem Verschlüsselungsprozess ein kryptografisches Commitment hinzu, das die verschlüsselten Daten an einen bestimmten Schlüssel bindet. Dadurch wird verhindert, dass ein Angreifer den verschlüsselten Datenschlüssel in der Befehlsdatei manipuliert und den Client veranlasst, Daten mit einem falschen Schlüssel zu entschlüsseln.

Ohne Schlüsselübergabe ist es für einen Angreifer möglich, den verschlüsselten Datenschlüssel in einer Befehlsdatei so zu ändern, dass er mit einem anderen Schlüssel entschlüsselt wird, was möglicherweise unbefugten Zugriff oder Datenbeschädigung ermöglicht. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`verhindert diesen Angriff, indem sichergestellt wird, dass der verschlüsselte Datenschlüssel nur mit dem ursprünglichen Schlüssel entschlüsselt werden kann, der bei der Verschlüsselung verwendet wurde.

Versionskompatibilität

Mit verschlüsselte Objekte `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` können nur mit V3-Implementierungen des S3-Verschlüsselungsclients und bestimmten Übergangsversionen von V2 entschlüsselt werden, die Unterstützung für das Lesen von V3-Formaten beinhalten. V2-Clients ohne diese Übergangsunterstützung können Befehlsdateien, die mit verschlüsselt wurden, nicht entschlüsseln. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

> **Warning**
>
> Bevor Sie die Verschlüsselung mit `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (mithilfe von `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` Commitment-Richtlinien) aktivieren, stellen Sie sicher, dass alle Clients, die Ihre verschlüsselten Objekte lesen müssen, auf Version 3 oder eine Übergangsversion aktualisiert wurden, die V3-Formate unterstützt. Wenn V2-Clients ohne Übergangsunterstützung versuchen, Objekte zu lesen, mit denen verschlüsselt wurde`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`, schlägt die Entschlüsselung fehl.

Während der Migration können Sie die `FORBID_ENCRYPT_ALLOW_DECRYPT` Commitment-Richtlinie verwenden, um weiterhin zu verschlüsseln, ohne `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` dass Ihre V3-Clients weiterhin Objekte lesen können, die mit Key Commitment verschlüsselt wurden. Dies bietet einen sicheren Migrationspfad, bei dem Sie zuerst alle Lesegeräte aktualisieren und dann zur Verschlüsselung mit Schlüsselzusage wechseln.

Aktualisieren Sie bestehende Clients, um neue Formate zu lesen

Der V3-Verschlüsselungsclient verwendet Verschlüsselungsalgorithmen und Key-Commitment-Funktionen, die V2-Clients standardmäßig nicht unterstützen. Der erste Schritt der Migration besteht darin, Ihre V2-Entschlüsselungsclients auf eine Version des AWS SDK for Ruby zu aktualisieren, die V3-verschlüsselte Objekte lesen kann. Nach Abschluss dieses Schritts können die V2-Clients Ihrer Anwendung Objekte entschlüsseln, die mit V3-Verschlüsselungsclients verschlüsselt wurden.

Um Objekte zu lesen, die von V3-Clients (die Richtlinien `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` Commitment-Richtlinien verwenden) verschlüsselt wurden, müssen Sie Version 1.93.0 oder höher des `aws-sdk-s3` Gems verwenden. Diese Version bietet Unterstützung für die Entschlüsselung von Objekten, die mit AES GCM mit Key Commitment verschlüsselt wurden.

Installation über die Befehlszeile

Verwenden Sie bei Projekten, die das `aws-sdk-s3` Gem über die Befehlszeile installieren, die Versionsoption, um zu überprüfen, ob die Mindestversion von 1.208.0 installiert ist.

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

Gemfiles verwenden

Für Projekte, die ein Gemfile zur Verwaltung von Abhängigkeiten verwenden, legen Sie die Mindestversion des `aws-sdk-s3` Gems auf 1.208.0 fest. Beispiel:

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. Ändern Sie Ihr Gemfile, um die Mindestversion anzugeben.
2. Führen Sie `bundle update aws-sdk-s3` den Befehl aus, um das Gem zu aktualisieren.
3. Führen Sie den Befehl aus, um Ihre Version zu überprüfen`bundle info aws-sdk-s3`.

> **Note**
>
> Nach dem Update auf die neueste Version können Ihre vorhandenen V2-Verschlüsselungsclients Objekte entschlüsseln, die von V3-Clients verschlüsselt wurden. Sie werden jedoch weiterhin neue Objekte mit V2-Algorithmen verschlüsseln, bis Sie sie, wie im nächsten Abschnitt beschrieben, auf V3 migrieren.

Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients zu V3

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 Schritte zeigen Ihnen, wie Sie Ihren Code erfolgreich von V2 auf V3 migrieren können.

Bevor Sie Ihren Code für die Verwendung des V3-Verschlüsselungsclients aktualisieren, stellen Sie sicher, dass Sie die vorherigen Schritte befolgt haben und die `aws-sdk-s3` Gem-Version 1.93.0 oder höher verwenden.

> **Note**
>
> Lesen Sie beim Entschlüsseln mit AES-GCM das gesamte Objekt bis zum Ende, bevor Sie die entschlüsselten Daten verwenden. Dadurch wird überprüft, ob das Objekt seit der Verschlüsselung nicht geändert wurde.

Konfiguration von V3-Clients

Der V3-Verschlüsselungsclient führt neue Konfigurationsoptionen ein, mit denen das Verhalten bei der Schlüsselübergabe und die Abwärtskompatibilität gesteuert werden können. Das Verständnis dieser Optionen ist für eine erfolgreiche Migration unerlässlich.

commitment_policy

Der `commitment_policy` Parameter steuert, wie der Verschlüsselungsclient bei Verschlüsselungs- und Entschlüsselungsvorgängen mit der Schlüsselzusage umgeht. Dies ist die wichtigste Konfigurationsoption für V3-Clients.

- `:require_encrypt_allow_decrypt`- Verschlüsselt neue Objekte mit Key Commitment und ermöglicht die Entschlüsselung von Objekten mit oder ohne Key Commitment. Dies ist die empfohlene Einstellung für die Migration, da sie mehr Sicherheit für neue Objekte bietet und gleichzeitig die Abwärtskompatibilität mit vorhandenen V2-Objekten gewährleistet.
- `:forbid_encrypt_allow_decrypt`- Verschlüsselt neue Objekte ohne Schlüsselbindung (mithilfe von V2-Algorithmen) und ermöglicht die Entschlüsselung von Objekten mit oder ohne Schlüsselbindung. Verwenden Sie diese Einstellung nur, wenn Sie das V2-Verschlüsselungsverhalten während der Migration beibehalten müssen, z. B. wenn einige Clients V3-verschlüsselte Objekte noch nicht lesen können.
- `:require_encrypt_require_decrypt`- Verschlüsselt neue Objekte mit Key Commitment und ermöglicht nur die Entschlüsselung von Objekten, die mit Key Commitment verschlüsselt wurden. Verwenden Sie diese Einstellung erst, nachdem alle Objekte mit Key Commitment erneut verschlüsselt wurden und alle Clients auf Version V3 aktualisiert wurden.

security_profile

Der `security_profile` Parameter bestimmt die Unterstützung für das Lesen von Objekten, die mit älteren Versionen des Verschlüsselungsclients geschrieben wurden. Dieser Parameter ist wichtig, um die Abwärtskompatibilität während der Migration aufrechtzuerhalten.

- `:v3_and_legacy`- Ermöglicht dem V3-Client, Objekte zu entschlüsseln, die mit V1- und V2-Verschlüsselungsclients verschlüsselt wurden. Verwenden Sie diese Einstellung während der Migration, um sicherzustellen, dass Ihre V3-Clients alle vorhandenen verschlüsselten Objekte lesen können.

- `:v3`— Ermöglicht dem V3-Client, nur Objekte zu entschlüsseln, die mit V2-Verschlüsselungsclients verschlüsselt wurden. Verwenden Sie diese Einstellung, wenn Sie bereits alle V1-Objekte in das V2-Format migriert haben.
- Wenn nicht angegeben, entschlüsselt der Client nur Objekte, die von V3-Clients verschlüsselt wurden. Verwenden Sie dies nur für die Entwicklung neuer Anwendungen, für die keine Legacy-Objekte vorhanden sind.

envelope_location

Der `envelope_location` Parameter bestimmt, wo Verschlüsselungsmetadaten (einschließlich des verschlüsselten Datenschlüssels) gespeichert werden. Dieser Parameter beeinflusst, welche Objekte durch AES GCM mit Key Commitment geschützt werden.

- `:metadata`(Standard) — Speichert Verschlüsselungsmetadaten in den Metadaten-Headern des S3-Objekts. Dies ist das Standardverhalten und wird für die meisten Anwendungsfälle empfohlen. Bei der Verwendung von Metadatenspeichern gilt AES GCM with Key Commitment nicht.
- `:instruction_file`- Speichert Verschlüsselungsmetadaten in einem separaten S3-Objekt (Befehlsdatei) mit einem konfigurierbaren Suffix. Bei der Verwendung von Befehlsdateien schützt AES GCM mit Key Commitment den verschlüsselten Datenschlüssel vor Manipulation. Verwenden Sie diese Einstellung, wenn Sie die zusätzliche Sicherheit benötigen, die Key Commitment für den Datenschlüssel selbst bietet.

Bei der Verwendung können Sie optional den `instruction_file_suffix` Parameter angeben`:instruction_file`, um das für Befehlsdateiobjekte verwendete Suffix anzupassen. Das Standardsuffix ist. `.instruction`

Wann sollten die einzelnen Konfigurationsoptionen verwendet werden

Folgen Sie während der Migration dieser empfohlenen Konfigurationsstrategie:

1. Erste Migration: Stellen Sie `commitment_policy: :require_encrypt_allow_decrypt` und ein`security_profile: :v3_and_legacy`. Auf diese Weise können Ihre V3-Clients neue Objekte mit Schlüsselbindung verschlüsseln und gleichzeitig alle vorhandenen V1- und V2-Objekte entschlüsseln.
2. Nach dem Upgrade aller Clients: Verwenden Sie weiterhin alle Objekte, für die ein Schlüsselverbindungsschutz erforderlich ist,

`commitment_policy: :require_encrypt_allow_decrypt` und zwar solange, `security_profile: :v3_and_legacy` bis Sie alle Objekte erneut verschlüsselt haben.

3. Vollständige V3-Erzwingung: Erst wenn alle Objekte mit Key Commitment erneut verschlüsselt wurden und Sie keine V1/V2-Objekte mehr lesen müssen, können Sie optional zu dem `security_profile` Parameter wechseln `commitment_policy: :require_encrypt_require_decrypt` und ihn entfernen (oder ihn auf „setzen“, `:v2` falls noch V2-Objekte existieren).

Verwenden Sie `envelope_location` also weiterhin Ihre bestehende Speichermethode (`:metadata`oder`:instruction_file`), es sei denn, Sie haben einen bestimmten Grund, sie zu ändern. Wenn Sie derzeit Metadatenspeicher verwenden und die zusätzliche Sicherheit von AES GCM mit Key Commitment für den Datenschlüssel nutzen möchten, können Sie zu dieser Option wechseln. Beachten Sie jedoch`:instruction_file`, dass dafür alle Clients aktualisiert werden müssen, die diese Objekte lesen.

Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients zu 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

Vor der Migration (V2)

```
require 'aws-sdk-s3'

# Create V2 encryption client with KMS
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy,
  commitment_policy: :forbid_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')
```

```
# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Während der Migration (V3 mit Abwärtskompatibilität)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Nach der Migration (V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3,
  # Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
  # This encrypts with key commitment and does not decrypt V2 objects
  commitment_policy: :require_encrypt_require_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
```

```
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Der Hauptunterschied in V3 besteht in der Hinzufügung des `commitment_policy` Parameters. Wenn Sie ihn auf einstellen, `:require_encrypt_require_decrypt` wird sichergestellt, dass neue Objekte mit Key Commitment verschlüsselt werden und dass der Client nur Objekte entschlüsselt, die mit Key Commitment verschlüsselt wurden, wodurch die Sicherheit vor Manipulation von Datenschlüsseln erhöht wird.

Der `put_object` Anruf selbst bleibt unverändert. Alle Sicherheitsverbesserungen werden auf Client-Ebene konfiguriert.

Weitere Beispiele

Dieser Abschnitt enthält zusätzliche Beispiele für spezifische Migrationsszenarien und Konfigurationsoptionen, die bei der Migration von V2 zu V3 nützlich sein können.

Befehlsdatei und Metadatenspeicher

Der S3-Verschlüsselungsclient kann Verschlüsselungsmetadaten (einschließlich des verschlüsselten Datenschlüssels) an zwei verschiedenen Orten speichern: in den Metadaten-Headern des S3-Objekts oder in einer separaten Befehlsdatei. Die Wahl der Speichermethode wirkt sich darauf aus, welche Objekte von AES GCM mit Key Commitment-Schutz profitieren.

Metadaten-Speicher (Standard)

Standardmäßig speichert der Verschlüsselungsclient Verschlüsselungsmetadaten in den Metadaten-Headern des S3-Objekts. Dies ist der empfohlene Ansatz für die meisten Anwendungsfälle, da er die Verschlüsselungsmetadaten zusammen mit dem Objekt behält und es nicht erforderlich ist, separate Befehlsdateiobjekte zu verwalten.

```
require 'aws-sdk-s3'

# Create V3 encryption client with metadata storage (default)
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :metadata # Explicitly set to metadata (this is the default)
```

```
)

# Encrypt and upload object
# Encryption metadata is stored in the object's metadata headers
client.put_object(bucket: 'my-bucket', key: 'my-object',body: 'secret data')
```

Bei Verwendung des Metadatenspeichers gilt AES GCM with Key Commitment nicht für den verschlüsselten Datenschlüssel. Die Inhaltsverschlüsselung profitiert jedoch immer noch von der Schlüsselzusage, wenn Sie `commitment_policy: :require_encrypt_allow_decrypt` oder `:require_encrypt_require_decrypt` verwenden.

Speicherung von Befehlsdateien

Alternativ können Sie den Verschlüsselungsclient so konfigurieren, dass Verschlüsselungsmetadaten in einem separaten S3-Objekt gespeichert werden, das als Befehlsdatei bezeichnet wird. Bei der Verwendung von Instruction Files mit V3 wird der verschlüsselte Datenschlüssel durch AES GCM mit Key Commitment geschützt, was zusätzliche Sicherheit vor Manipulation von Datenschlüsseln bietet.

```
require 'aws-sdk-s3'

# Create V3 encryption client with instruction file storage
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :instruction_file, # Store metadata in separate instruction file
  instruction_file_suffix: '.instruction' # Optional: customize the suffix (default is
 '.instruction')
)

# Encrypt and upload object
# Encryption metadata is stored in a separate object: 'my-object.instruction'
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# When retrieving the object, the client automatically reads the instruction file
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Bei der Verwendung `envelope_location: :instruction_file` erstellt der Verschlüsselungsclient zwei S3-Objekte:

1. Das verschlüsselte Datenobjekt `my-object` (z. B.
2. Die Anweisungsdatei, die Verschlüsselungsmetadaten enthält `my-object.instruction` (z. B.

Mit dem `instruction_file_suffix` Parameter können Sie das für Befehlsdateien verwendete Suffix anpassen. Der Standardwert ist `.instruction`.

Wann sollten die einzelnen Speichermethoden verwendet werden

- Verwenden Sie den Metadatenspeicher für die meisten Szenarien. Es vereinfacht die Objektverwaltung, da Verschlüsselungsmetadaten zusammen mit dem Objekt übertragen werden.
- Verwenden Sie Instruction File Storage, wenn die Größe der Objektmetadaten ein Problem darstellt oder wenn Sie Verschlüsselungsmetadaten vom verschlüsselten Objekt trennen müssen. Beachten Sie, dass für die Verwendung von Befehlsdateien zwei S3-Objekte (das verschlüsselte Objekt und die zugehörige Befehlsdatei) statt eines verwaltet werden müssen.

> **Warning**
>
> Wenn Sie vom Metadatenspeicher zum Speicher für Befehlsdateien wechseln (oder umgekehrt), können vorhandene Objekte, die mit der alten Speichermethode verschlüsselt wurden, von Clients, die mit der neuen Speichermethode konfiguriert wurden, nicht gelesen werden. Planen Sie Ihre Speichermethode sorgfältig und sorgen Sie für Konsistenz in Ihrer gesamten Anwendung.

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Handbuch beschrieben. Für Benachrichtigungen über Aktualisierungen dieser Dokumentation können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Reorganisation der Inhalte	Aktualisierung des Inhaltsverzeichnisses und der Organisation der Inhalte, um sie besser an andere AWS SDKs anzupassen.	29. März 2025
Beobachtbarkeit	Es wurden Informationen zur Ruby-Observability hinzugefügt.	24. Januar 2025
Allgemeine Aktualisierungen	Die mindestens erforderliche Ruby-Version wurde auf v2.5 aktualisiert. Die Links zu den Ressourcen wurden aktualisiert.	13. November 2024
Inhaltsverzeichnis und angeleitete Beispiele	Geführte Beispiele wurden entfernt, um auf das umfassendere Codebeispiel-Repository verweisen zu können.	10. Juli 2024
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	8. Mai 2023

	Bewährte Sicherheitsmethoden in IAM. Aktualisierungen für Erste Schritte.	
Allgemeine Updates	Aktualisierung der Willkommensseite für relevante externe Ressourcen. Außerdem wurde die mindestens erforderliche Ruby-Version für v2.3 aktualisiert. Die AWS Key Management Service Abschnitte wurden aktualisiert, um Terminologieaktualisierungen Rechnung zu tragen. Die Nutzungsinformationen des REPL-Dienstprogramms wurden aus Gründen der Übersichtlichkeit aktualisiert.	08. August 2022
Korrigieren defekter Links	Fehlerhafte Beispiel-Links wurden behoben. Überflüssige Seite mit Tipps und Tricks entfernt; Weiterleitung zu EC2 Amazon-Beispielinhalten. Es wurden Listen der Codebeispiele hinzugefügt, die GitHub im Codebeispiel-Repository verfügbar sind.	03. August 2022
SDK-Metriken	Informationen zur Aktivierung von SDK-Metriken für Enterprise Support, die veraltet sind, wurden entfernt.	28. Januar 2022

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.