

デベロッパーガイド

AWS SDK for Ruby



AWS SDK for Ruby: デベロッパーガイド

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

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

AWS SDK for Ruby とは	1
その他のドキュメントとリソース	1
AWS クラウドへのデプロイ	1
SDK メジャーバージョンのメンテナンスとサポート	2
入門	3
による認証 AWS	3
AWS アクセスポータルセッションを開始する	4
詳細認証情報	5
SDK のインストール	6
前提条件	6
SDK のインストール	6
シンプルなアプリケーションの作成	7
コードの記述	7
プログラムの実行	8
Windows ユーザー向けの注意事項	9
次のステップ	9
設定	10
設定の優先順位	10
AWS リージョン	11
解決のためのリージョン検索順序	11
リージョンの設定方法	12
環境変数	13
AWS SDK for Ruby 環境変数	14
認証情報プロバイダ	14
認証情報プロバイダーチェーン	14
認証情報プロバイダーチェーン	14
AWS STS アクセストークンの作成	16
SDK 設定	17
Aws.config	17
サービスクライアント	18
再試行	19
コードでのクライアントの再試行動作の指定	19
オブザーバビリティ	20
サービスクライアントの OTelProvider の設定	20

OTelProvider すべてのサービスクライアントの の設定	23
カスタムテレメトリプロバイダーの設定	24
スパン属性	24
HTTP	26
標準外のエンドポイントを設定する	26
SDK の使用	27
REPL ユーティリティの使用	27
前提条件	27
バンドラーの設定	27
REPL の実行	28
Ruby on Rails で SDK を使用する	28
クライアントからのワイヤトレースを使用したデバッグ	29
スタブによるテスト	30
クライアントレスポンスをスタブする	30
クライアントエラーをスタブする	32
ページ分割	32
ページ分割レスポンスは列挙可能です	32
ページ分割レスポンスを手動で処理	33
ページ分割データクラス	33
ウェイター	33
ウェーターの呼び出し	34
待機の失敗	34
ウェーターの設定	35
ウェーターの拡張	35
コードの例	37
Aurora	38
Auto Scaling	39
CloudTrail	41
アクション	41
CloudWatch	45
アクション	41
Amazon Cognito Identity Provider	58
Amazon Comprehend	59
シナリオ	60
Amazon DocumentDB	61
サーバーレスサンプル	61

DynamoDB	62
基本	64
アクション	41
シナリオ	60
サーバーレスサンプル	61
Amazon EC2	91
アクション	41
Elastic Beanstalk	125
アクション	41
EventBridge	131
シナリオ	60
AWS Glue	149
基本	64
アクション	41
IAM	177
基本	64
アクション	41
Kinesis	235
サーバーレスサンプル	61
AWS KMS	238
アクション	41
Lambda	242
基本	64
アクション	41
シナリオ	60
サーバーレスサンプル	61
Amazon MSK	272
サーバーレスサンプル	61
Amazon Polly	273
アクション	41
シナリオ	60
Amazon RDS	278
アクション	41
サーバーレスサンプル	61
Amazon S3	285
基本	64

アクション	41
シナリオ	60
サーバーレスサンプル	61
Amazon SES	317
アクション	41
Amazon SES API v2	323
アクション	41
Amazon SNS	324
アクション	41
サーバーレスサンプル	61
Amazon SQS	334
アクション	41
サーバーレスサンプル	61
AWS STS	348
アクション	41
Amazon Textract	349
シナリオ	60
Amazon Translate	350
シナリオ	60
Amazon WorkDocs	352
アクション	41
バージョンの移行	355
サイドバイサイドの使用	355
一般的な違い	355
クライアントの違い	356
リソースの違い	357
セキュリティ	359
データ保護	359
Identity and Access Management	360
コンプライアンス検証	361
耐障害性	362
インフラストラクチャセキュリティ	362
最小 TLS バージョンの適用	363
OpenSSL バージョンの確認	363
TLS サポートのアップグレード	364
S3 暗号化クライアントの移行	364

移行の概要	364
新しいフォーマットを読み取るために既存のクライアントを更新する	364
暗号化および復号化クライアントを V2 に移行する	366
ドキュメント履歴	370
.....	ccclxxii

AWS SDK for Ruby とは

AWS SDK for Ruby デベロッパーガイドへようこそ。AWS SDK for Ruby は、Amazon Simple Storage Service (Amazon S3) AWS のサービス、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon DynamoDB など、ほぼすべての のサポートライブラリを提供します。

AWS SDK for Ruby デベロッパーガイドでは、AWS SDK for Ruby をインストール、セットアップ、使用して、 を使用する Ruby アプリケーションを作成する方法について説明します AWS のサービス。

[AWS SDK for Ruby の開始方法](#)

その他のドキュメントとリソース

AWS SDK for Ruby 開発者向けのその他のリソースについては、以下を参照してください。

- [AWS SDKsおよびツールリファレンスガイド](#) – AWS SDKs
- [AWS SDK for Ruby API リファレンス - バージョン 3](#)
- GitHub [AWS のコード例リポジトリ](#)
- [RubyGems.org](#) — SDK の最新バージョンは、サービス固有の gem にモジュール化されています。こちらから入手できます。
 - [サポートされているサービス](#) – AWS SDK for Ruby がサポートするすべての Gem を一覧表示します。
- AWS GitHub の SDK for Ruby ソース :
 - [ソースと README](#)
 - [各 Gem の変更ログ](#)
 - [v2 から v3 への移行](#)
 - [問題点](#)
 - [主要アップグレードに関する注意事項](#)
- [開発者ブログ](#)

AWS クラウドへのデプロイ

AWS Elastic Beanstalk や AWS のサービス などの を使用して AWS CodeDeploy 、アプリケーションを AWS クラウドにデプロイできます。Elastic Beanstalk で Ruby アプリケーションをデプロイ

するには、AWS Elastic Beanstalk デベロッパーガイドの「[EB CLI と Git を使用した Ruby での Elastic Beanstalk アプリケーションのデプロイ](#)」を参照してください。AWS デプロイサービスの概要については、[AWSのデプロイオプションの概要](#)を参照してください。

SDK メジャーバージョンのメンテナンスとサポート

SDK メジャーバージョンのメンテナンスとサポート、およびその基礎的な依存関係については、[AWS SDK とツール共有設定および認証情報リファレンスガイド](#)で以下を参照してください。

- [AWS SDKsメンテナンスポリシー](#)
- [AWS SDKsとツールのバージョンサポートマトリックス](#)

AWS SDK for Ruby の開始方法

SDK をインストール、セットアップ、使用して、Ruby アプリケーションを作成して AWS リソースにプログラムでアクセスする方法について説明します。

トピック

- [を使用した AWS SDK for Ruby の認証 AWS](#)
- [AWS SDK for Ruby のインストール](#)
- [AWS SDK for Ruby を使用したシンプルなアプリケーションの作成](#)

を使用した AWS SDK for Ruby の認証 AWS

を使用して開発 AWS するときに、コードが で認証される方法を確認する必要があります AWS のサービス。AWS リソースへのプログラムによるアクセスは、環境と利用可能な AWS アクセスに応じてさまざまな方法で設定できます。

認証方法を選択して SDK 用に設定するには、AWS SDK とツールのリファレンスガイドの「[認証とアクセス](#)」を参照してください。

ローカルで開発中で、雇用主から認証方法が与えられていない新規ユーザーは、を設定することをお勧めします AWS IAM Identity Center。この方法には、設定を簡単に行ったり、AWS アクセスポータルに定期的にサインインしたりするための AWS CLI をインストールすることもあります。

この方法を選択した場合は、「SDK およびツールリファレンスガイド」の「[IAM Identity Center 認証](#)」の手順を完了します。AWS SDKs その後、環境には次の要素が含まれている必要があります。

- アプリケーションを実行する前に AWS アクセスポータルセッションを開始 AWS CLI するために使用する。
- SDK から参照できる設定値のセットを含む [default] プロファイルがある[共有 AWSconfig ファイル](#)。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「[共有ファイルの場所](#)」を参照してください。
- 共有 config ファイルは [region](#) 設定を設定します。これにより、SDK AWS リージョンが AWS リクエストに使用するデフォルトが設定されます。このリージョンは、使用するリージョンが指定されていない SDK サービスリクエストに使用されます。
- SDK は、リクエストを AWS に送信する前に、プロファイルの [SSO トークンプロバイダー設定](#)を使用して認証情報を取得します。sso_role_name 値は、IAM Identity Center アクセス許可セット

に接続された IAM ロールであり、アプリケーションで AWS のサービス 使用されている へのアクセスを許可します。

次のサンプル config ファイルは、SSO トークンプロバイダー設定で設定されたデフォルトプロファイルを示しています。プロファイルの `sso_session` 設定は、指定された [sso-session セクション](#)を参照します。sso-session セクションには、AWS アクセスポータルセッションを開始するための設定が含まれています。

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

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

AWS SDK for Ruby では、IAM Identity Center 認証を使用するアプリケーションに追加のパッケージ (SSOや などSSOIDC) は必要ありません。

AWS アクセスポータルセッションを開始する

にアクセスするアプリケーションを実行する前に AWS のサービス、SDK が IAM Identity Center 認証を使用して認証情報を解決するためのアクティブな AWS アクセスポータルセッションが必要です。設定したセッションの長さによっては、アクセスが最終的に期限切れになり、SDK で認証エラーが発生します。AWS アクセスポータルにサインインするには、で次のコマンドを実行します AWS CLI。

```
aws sso login
```

ガイダンスに従い、デフォルトのプロファイルを設定している場合は、`--profile` オプションを指定してコマンドを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルを使用している場合、コマンドは `aws sso login --profile named-profile` です。

既にアクティブなセッションがあるかどうかをオプションでテストするには、次の AWS CLI コマンドを実行します。

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

セッションがアクティブな場合、このコマンドへの応答により、共有 config ファイルに設定されている IAM Identity Center アカウントとアクセス許可のセットが報告されます。

Note

既にアクティブな AWS アクセスポータルセッションがあり、 を実行している場合は `aws sso login`、認証情報を指定する必要はありません。

サインインプロセスでは、データ AWS CLI へのアクセスを許可するように求められる場合があります。AWS CLI は SDK for Python 上に構築されているため、アクセス許可メッセージには `botocore` 名前のバリエーションが含まれている可能性があります。

詳細認証情報

人間のユーザーとは、別名人間 ID と呼ばれ、人、管理者、デベロッパー、オペレーター、およびアプリケーションのコンシューマーを指します。AWS 環境とアプリケーションにアクセスするには、ID が必要です。組織のメンバーである人間のユーザー、つまり、ユーザーや開発者は、ワークフォースアイデンティティと呼ばれます。

アクセス時に一時的な認証情報を使用します AWS。一時的な認証情報を提供するロールを引き受けることで、人間のユーザーの ID プロバイダーを使用して AWS アカウントへのフェデレーションアクセスを提供できます。一元的なアクセス管理を行うには、AWS IAM Identity Center (IAM Identity Center) を使用して、ご自分のアカウントへのアクセスと、それらのアカウント内でのアクセス許可を管理することをお勧めします。その他の代替案については、以下を参照してください。

- ベストプラクティスの詳細については、IAM ユーザーガイドの「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。
- 短期 AWS 認証情報を作成するには、IAM ユーザーガイドの「[一時的なセキュリティ認証情報](#)」を参照してください。
- AWS SDK for Ruby 認証情報プロバイダーチェーン、および SDK によって一連の認証方法が自動的に試行される方法については、「」を参照してください [認証情報プロバイダーチェーン](#)。
- AWS SDK 認証情報プロバイダーの設定については、「SDK とツールリファレンスガイド」の「[標準化された認証情報プロバイダー](#)」を参照してください。AWS SDKs

AWS SDK for Ruby のインストール

このセクションには、AWS SDK for Ruby の前提条件とインストール手順が含まれています。

前提条件

AWS SDK for Ruby を使用する前に、で認証する必要があります AWS。認証の設定の詳細については、「[を使用した AWS SDK for Ruby の認証 AWS](#)」を参照してください。

SDK のインストール

AWS SDK for Ruby は、Ruby Gem と同様にインストールできます。gem は [RubyGems](#) で購入できます。AWS SDK for Ruby はモジュール式に設計されており、で区切られています AWS のサービス。aws-sdkgem 全体のインストールはサイズが大きく、1 時間以上かかる場合があります。

AWS のサービス 使用する にのみ Gem をインストールすることをお勧めします。これらは のような名前aws-sdk-*service_abbreviation*で、完全なリストは AWS SDK for Ruby README ファイルの [Supported Services](#) テーブルにあります。たとえば、Amazon S3 サービスとインターフェースするための gem は、[aws-sdk-s3](#) で直接入手できます。

Ruby バージョンマネージャー (RVM)

システム Ruby を使用する代わりに、次のような Ruby バージョンマネージャーの使用をお勧めします。

- [RVM](#)
- [chruby](#)
- [rbenv](#)

例えば、Amazon Linux 2 オペレーティングシステムを使用している場合、次のコマンドを使用して RVM を更新し、使用可能な Ruby バージョンを一覧表示してから、AWS SDK for Ruby で開発に使用するバージョンを選択できます。Ruby の最小必須バージョンは 2.5 です。

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

バンドラー

[Bundler](#) を使用する場合、次のコマンドは Amazon S3 用の AWS SDK for Ruby Gem をインストールします。

1. バンドラーをインストールしてGemfileを作成します。

```
$ gem install bundler
$ bundle init
```

2. 作成した を開きGemfile、コードが使用する AWS サービス Gem ごとにgem行を追加します。Amazon S3 の例に従って作業する場合は、ファイルの末尾に次のテキスト行を追加します。

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

3. Gemfile を保存する
4. お使いのGemfileで指定されている依存関係をインストールします。

```
$ bundle install
```

AWS SDK for Ruby を使用したシンプルなアプリケーションの作成

AWS SDK for Ruby を使用して Amazon S3 に挨拶します。次の例では、すべての Amazon S3 バケットのリストを表示します。

コードの記述

次のコードをコピーして、新しいソースファイルに貼り付けます。ファイルを `hello-s3.rb` と名付けます。

```
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
end
```

```
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 SDK for Ruby はモジュール式に設計されており、で区切られています AWS のサービス。gem をインストールすると、Ruby ソースファイルの先頭にある `require` ステートメントで、Amazon S3 サービス用の AWS SDK クラスとメソッドがインポートされます。利用可能な AWS サービス Gem の完全なリストについては、AWS SDK for Ruby README ファイルの「[サポートされているサービス](#)」表を参照してください。

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

プログラムの実行

Ruby プログラムを実行するには、コマンドプロンプトを開きます。Ruby プログラムを実行する一般的なコマンド構文は次のとおりです。

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

このサンプルコードでは引数を使用しません。このコードを実行するには、コマンドプロンプトで以下を入力します。

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

Windows ユーザー向けの注意事項

Windows の SSL 証明書で Ruby コードを実行すると、次のようなエラーが表示されることがあります。

```
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'
...

```

この問題を修正するには、最初の AWS 呼び出しの前のどこかで、Ruby ソースファイルに次の行を追加します。

```
Aws.use_bundled_cert!
```

Ruby プログラムで `aws-sdk-s3` gem のみを使用している場合は、バンドルされた証明書を使用するために `aws-sdk-core` gem を追加する必要があります。

次のステップ

その他の多くの Amazon S3 オペレーションをテストするには、GitHub で [AWS のコードサンプルリポジトリ](#)を確認してください。

AWS SDK for Ruby の設定

AWS SDK for Ruby を設定する方法について説明します。で開発 AWS するときに、コードが で認証される方法を確認する必要があります AWS のサービス。AWS リージョン 使用する も設定する必要があります。

[AWS SDKs およびツールリファレンスガイド](#)には、多くの AWS SDKs で一般的な設定、機能、その他の基本的な概念も含まれています。

トピック

- [設定の優先順位](#)
- [AWS SDK for Ruby AWS リージョン の の設定](#)
- [AWS SDK for Ruby の環境変数およびその他の設定オプション](#)
- [AWS SDK for Ruby 認証情報プロバイダーの使用](#)
- [AWS SDK for Ruby Aws.configでの を使用した一般的な設定](#)
- [AWS SDK for Ruby でのサービスクライアントの作成](#)
- [AWS SDK for Ruby での再試行の設定](#)
- [AWS SDK for Ruby でのオブザーバビリティ機能の設定](#)
- [AWS SDK for Ruby 内での HTTP レベルの設定の構成](#)

[共有 ファイルconfigと credentialsファイル](#)は、設定に使用できます。すべての AWS SDK 設定については、「SDK およびツールリファレンスガイド」の「[設定リファレンス](#)」を参照してください。AWS SDKs

異なるプロファイルを使用して、異なる設定を保存できます。SDK がロードするアクティブなプロファイルを指定するには、AWS_PROFILE環境変数または profileのオプションを使用できます `Aws.config`。

設定の優先順位

グローバル設定は、ほとんどの SDK でサポートされ、AWS のサービス全体に幅広く影響する機能、認証情報プロバイダー、およびその他の機能を設定します。すべての AWS SDKs には、グローバル設定の値を見つけるためにチェックする一連の場所 (またはソース) があります。すべての設定がすべてのソースで使用できるわけではありません。設定検索の優先順位は次のとおりです。

1. コードまたはサービスクライアント自体に設定されている明示的な設定は、他の設定よりも優先されます。
 - a. クライアントコンストラクタに直接渡されるパラメータは、最も優先されます。
 - b. `Aws.config` は、グローバル設定またはサービス固有の設定をチェックします。
2. 環境変数が確認されます。
3. 共有 AWS credentials ファイルがチェックされます。
4. 共有 AWS config ファイルがチェックされます。
5. AWS SDK for Ruby ソースコード自体によって提供されるデフォルト値が最後に使用されます。

AWS SDK for Ruby AWS リージョン の の設定

を使用して、特定の地域 AWS のサービス で動作する にアクセスできます AWS リージョン。これは、冗長性と、ユーザーとユーザーがアクセスする場所の近くでデータとアプリケーションを実行し続けるために役立ちます。

Important

ほとんどのリソースは特定の に存在し AWS リージョン、SDK を使用するときリソースに正しいリージョンを指定する必要があります。

AWS リクエストに使用する SDK for Ruby AWS リージョン のデフォルトを設定する必要があります。このデフォルトは、リージョンで指定されていない SDK サービスメソッド呼び出しに使用されます。

region設定の詳細については、『AWS SDK とツールのリファレンスガイド』の「[AWS リージョン](#)」を参照のこと。これには、共有 AWS config ファイルまたは環境変数を使用してデフォルトのリージョンを設定する方法の例も含まれています。

解決のためのリージョン検索順序

AWS のサービスのほとんどのサービスでは、使用時にリージョンを設定する必要があります。AWS SDK for Ruby は、次の順序でリージョンを検索します。

1. クライアントまたはリソースオブジェクト内でのリージョンの設定
2. を使用したリージョンの設定 `Aws.config`

3. 環境変数でリージョンを設定する
4. 共有configファイルを使用したリージョンの設定

リージョンの設定方法

このセクションでは、最も一般的なアプローチから始めて、リージョンを設定するさまざまな方法について説明します。

共有 **config** ファイルでリージョンを設定する。

共有 AWS config ファイルで `region` 変数を設定して、リージョンを設定します。共有 config ファイルの使用についての詳細は、「[AWS SDK とツールのリファレンスガイド](#)」の「[共有設定ファイルおよび認証情報ファイル](#)」を参照してください。

config ファイルにこの値を設定する例を以下に示します。

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

AWS_SDK_CONFIG_OPT_OUT 環境変数が設定されている場合、共有 config ファイルはチェックされません。

環境変数でリージョンを設定する

AWS_REGION 環境変数を設定して、リージョンを設定します。

Linux または macOS のような Unix ベースのシステムでは、`export` コマンドでこの変数を設定します。次の例では、リージョンを `us-west-2` に設定します。

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

Windows でこれらの変数を設定するには、`set` コマンドを使用します。次の例では、リージョンを `us-west-2` に設定します。

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

Aws.config でのリージョンの設定

`region` 値を `Aws.config` ハッシュに追加してリージョンを設定します。次の例は `Aws.config` ハッシュを更新して `us-west-1` リージョンを使用します。

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

以後に作成するすべてのクライアントやリソースは、このリージョンにバインドされます。

クライアントまたはリソースオブジェクト内でのリージョンの設定

AWS クライアントまたはリソースを作成するときにリージョンを設定します。次の例では、us-west-1 リージョンに Amazon S3 リソースオブジェクトを作成します。AWS リソースに適したリージョンを選択します。サービスクライアントオブジェクトはイミュータブルであるため、リクエストを実行する各サービスについて、または異なった設定を使用する同じサービスにリクエストを実行するために、新しいクライアントを作成する必要があります。

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

AWS SDK for Ruby の環境変数およびその他の設定オプション

多くの設定はコードの外部で処理できます。ほとんどの設定は、環境変数として設定することも、別の共有 AWS config ファイルで設定することもできます。共有 config ファイルは、プロファイルと呼ばれる設定の個別のセットを維持し、異なる環境やテストに異なる設定を提供できます。

環境変数と共有 config ファイル設定は、SDKs とツール間で AWS 標準化および共有され、さまざまな言語間で一貫した機能をサポートします。

これらのメソッドを使用したアプリケーションの設定と、各クロス SDK 設定の詳細については、AWS SDKs およびツールリファレンスガイドを参照してください。対象となるページには、次のようなものがあります。

- [共有 config ファイルと credentials ファイル](#) - プロファイルとこれらの設定ファイルの形式について説明します。
- [共有ファイル config と credentials ファイルの場所](#) - これらのファイルのデフォルトの場所と変更方法について説明します。
- [環境変数のサポート](#) - 環境変数の設定について説明します。
- [リファレンスの設定](#) - すべてのクロス SDK 設定のリファレンス情報。このリファレンスには、使用可能なすべての [環境変数](#) と共有 [config ファイル設定が含まれています](#)。

AWS SDK for Ruby 環境変数

AWS SDK for Ruby は、ほとんどの AWS SDKs でサポートされているクロス SDK 環境変数以外にも、いくつかの一意の環境変数をサポートしています。

AWS_SDK_CONFIG_OPT_OUT

AWS SDK for Ruby 環境変数 `AWS_SDK_CONFIG_OPT_OUT` が設定されている場合、共有 AWS config ファイルは、通常は `~/.aws/config`、設定値には使用されません。

AMAZON_REGION

を設定 `AWS_REGION` するための の代替環境変数 AWS リージョン。この値は、`AWS_REGION` が使用されていない場合にのみチェックされます。

AWS SDK for Ruby 認証情報プロバイダーの使用

へのすべてのリクエストは、が発行した認証情報を使用して暗号で署名 AWS する必要があります AWS。実行時に、SDK は複数の場所をチェックして認証情報の設定値を取得します。

を使用した認証 AWS は、コードベース外で処理できます。認証情報プロバイダーチェーンを使用して、SDK によって多くの認証方法を自動的に検出、使用、更新できます。

プロジェクトの AWS 認証を開始するためのガイド付きオプションについては、SDK およびツール リファレンスガイドの「[認証とアクセス](#)」を参照してください。AWS SDKs

認証情報プロバイダーチェーン

クライアントの構築時に認証情報プロバイダーを明示的に指定しない場合、AWS SDK for Ruby は認証情報プロバイダーチェーンを使用して、認証情報を提供できる一連の場所をチェックします。SDK がこれらの場所のいずれかで認証情報を検出すると、検索は停止します。

認証情報プロバイダーチェーン

すべての SDK には、AWS のサービスに対するリクエストに使用する有効な認証情報を取得するためにチェックする一連の場所 (またはソース) があります。有効な認証情報が見つかったら、検索は停止されます。この体系的な検索は、デフォルトの認証情報プロバイダーチェーンと呼ばれます。

Note

新規ユーザーに推奨されるアプローチで開始した場合は、「使用開始」のトピックの [を使用した AWS SDK for Ruby の認証 AWS](#) 中に AWS IAM Identity Center 認証を設定します。その他の認証方法もさまざまな状況で役に立ちます。セキュリティリスクを避けるため、常に短期の認証情報を使用することをお勧めします。その他の認証方法については、『AWS SDK とツールのリファレンスガイド』の「[認証とアクセス](#)」を参照してください。

チェーンのステップごとに、値を設定するさまざまな方法があります。コード内で直接値を設定することが常に優先され、次に環境変数として `を` 設定し、次に共有 AWS config ファイルで `を` 設定します。

AWS SDKs およびツールリファレンスガイドには、すべての SDK および `で` 使用される AWS SDKs 設定に関する情報が記載されています AWS CLI。共有 AWS config ファイルを使用して SDK を設定する方法の詳細については、「[共有設定ファイルと認証情報ファイル](#)」を参照してください。環境変数を設定して SDK を設定する方法の詳細については、「[環境変数のサポート](#)」を参照してください。

で認証するために AWS、AWS SDK for Ruby は、次の表に示す順序で認証情報プロバイダーをチェックします。

認証情報プロバイダー (優先順位順)	AWS SDKs とツールのリファレンスガイド	AWS SDK for Ruby API リファレンス
AWS アクセスキー (一時的および長期的な認証情報)	AWS アクセスキー	Aws::Credentials Aws::SharedCredentials
AWS Security Token Service (AWS STS) からのウェブ ID トークン	ロール認証情報プロバイダーを引き受けます role_arn、role_session_name、および web_identity_token_file の使用	Aws::AssumeRoleWebIdentityCredentials

認証情報プロバイダー (優先順位順)	AWS SDKsとツールのリファレンスガイド	AWS SDK for Ruby API リファレンス
AWS IAM Identity Center。 このガイドでは、 を使用した AWS SDK for Ruby の認証 AWS を参照してください。	IAM Identity Center 認証情報プロバイダー	Aws::SSOCredentials
信頼されたエンティティプロバイダー (AWS_ROLE_ARN など) このガイドでは、「 AWS STS アクセストークンの作成 」を参照してください。	ロール認証情報プロバイダーを引き受けます role_arn および role_session_name を使用する	Aws::AssumeRoleCredentials
プロセス認証情報プロバイダー	プロセス認証情報プロバイダー	Aws::ProcessCredentials
Amazon Elastic Container Service (Amazon ECS) の認証情報。	コンテナ認証情報プロバイダー	Aws::ECSCredentials
Amazon Elastic Compute Cloud (Amazon EC2) インスタンスプロファイル認証情報 (IMDS 認証情報プロバイダー)	IMDS 認証情報プロバイダー	Aws::InstanceProfileCredentials

AWS SDK for Ruby 環境変数AWS_SDK_CONFIG_OPT_OUTが設定されている場合、共有 AWS configファイルは通常 にあり ~/.aws/config、認証情報を解析しません。

AWS STS アクセストークンの作成

ロールを引き受けるには、通常アクセスできない AWS リソースにアクセスするために使用できる一時的なセキュリティ認証情報のセットを使用します。これらの一時的な認証情報は、アクセスキー ID、シークレットアクセスキー、およびセキュリティトークンで構成されています。[Aws::AssumeRoleCredentials](#) メソッドを使用して AWS Security Token Service (AWS STS) アクセストークンを作成できます。

次の例では、`linked::account::arn` が引き受けるロールの Amazon リソースネーム (ARN) で、`session-name` が引き受けたロールセッションの識別子である場合、アクセストークンを使用して Amazon S3 クライアントオブジェクトを作成します。

```
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)
```

`role_arn` または `role_session_name` または代わりに共有 AWS config ファイルを使用した設定の詳細については、「[SDK およびツールリファレンスガイド](#)」の「[ロール認証情報プロバイダーを引き受ける](#)」を参照してください。AWS SDKs

AWS SDK for Ruby **Aws.config**での を使用した一般的な設定

Aws.config

すべての AWS クラスのコード内でグローバル設定を提供するには、gem で[Aws.config](#)利用可能な `aws-sdk-core` を使用します。

`Aws.config` は、さまざまな用途で 2 つの構文をサポートしています。グローバル設定は、すべてまたは AWS のサービス 特定のサービスに適用できます。サポートされている設定の完全なリストについては、AWS SDK for Ruby API リファレンスの `ClientOptions`「」を参照してください。

によるグローバル設定 **Aws.config**

を使用してサービスに依存しない設定を行うには `Aws.config`、次の構文を使用します。

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

これらの設定は、作成されたサービスクライアントにマージされます。

グローバル設定の例：

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

グローバルにサポートされていない設定名を使用しようとすると、サポートされていないタイプのサービスのインスタンスを作成しようとするとエラーが発生します。この場合、代わりにサービス固有の構文を使用します。

によるサービス固有の設定 **Aws.config**

を使用してサービス固有の設定を行うには `Aws.config`、次の構文を使用します。

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

これらの設定は、そのサービスタイプの作成されたすべてのサービスクライアントにマージされます。

Amazon S3 にのみ適用される設定の例：

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

は、対応する [AWS SDK for Ruby gem 名](#)を確認し、「」の後に続くサフィックスを使用して識別 `<service identifier>` できます `aws-sdk-`。例：

- の場合 `aws-sdk-s3`、サービス識別子文字列は `s3` 「」です。
- の場合 `aws-sdk-ecs`、サービス ID 文字列は `ecs` 「」です。

AWS SDK for Ruby でのサービスクライアントの作成

にリクエストを行うには AWS のサービス、まずそのサービスのクライアントをインスタンス化します。タイムアウト、HTTP クライアント、再試行設定など、サービスクライアントの共通設定を設定できます。

各サービスクライアントには、AWS リージョン と認証情報プロバイダーが必要です。SDK はこれらの値を使用して、リソースの正しいリージョンにリクエストを送信し、正しい認証情報を使用してリクエストに署名します。これらの値は、プログラムでコードで指定することも、環境から自動的にロードすることもできます。

- クライアントクラスをインスタンス化するときは、AWS 認証情報を指定する必要があります。SDK が認証プロバイダーをチェックする順序については、「」を参照してください [認証情報プロバイダーチェーン](#)。

- SDK には、設定の値を見つけるためにチェックする一連の場所 (またはソース) があります。詳細については、「[設定の優先順位](#)」を参照してください。

SDK for Ruby には、AWS のサービスへのインターフェイスを提供するクライアントクラスが含まれています。各クライアントクラスは特定の をサポートし AWS のサービス、 の規則に従います `Aws::<service identifier>::Client`。たとえば、[Aws::S3::Client](#) は Amazon Simple Storage Service サービスへのインターフェイスを提供し、[Aws::SQS::Client](#) は Amazon Simple Queue Service サービスへのインターフェイス [Aws::SQS::Client](#) を提供します。

すべての のすべてのクライアントクラス AWS のサービス はスレッドセーフです。

設定オプションは、クライアントコンストラクタとリソースコンストラクタに直接渡すことができます。これらのオプションは、環境と `Aws.config` デフォルトよりも優先されます。

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

AWS SDK for Ruby での再試行の設定

AWS SDK for Ruby には、デフォルトの再試行動作とカスタマイズ可能な設定オプションが用意されています。を呼び出すと、予期しない例外が返される AWS のサービス ことがあります。呼び出しが再試行されると、スロットリングや一時的なエラーなど、特定のタイプのエラーが成功する可能性があります。

再試行動作は、共有 AWS config ファイルの環境変数または設定を使用してグローバルに設定できます。このアプローチの詳細については、SDK およびツールリファレンスガイドの「[再試行動作](#)」を参照してください。AWS SDKs また、再試行戦略の実装に関する詳細情報と、別の方法を選択する方法も含まれています。

または、次のセクションに示すように、これらのオプションをコードで設定することもできます。

コードでのクライアントの再試行動作の指定

デフォルトでは、AWS SDK for Ruby は最大 3 回の再試行を実行し、再試行間隔は 15 秒、合計 4 回の再試行を実行します。したがって、オペレーションがタイムアウトするまでに最大 60 秒かかることがあります。

以下の例では、us-west-2 リージョンで Amazon S3 クライアントを作成し、クライアントオペレーションの再試行間隔を 5 秒、最大再試行回数を 2 回に設定しています。したがって、Amazon S3 クライアントオペレーションがタイムアウトするまでに最大 15 秒かかることがあります。

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

コードまたはサービスクライアント自体に設定された明示的な設定セットは、環境変数や共有 config ファイルに設定された設定より優先します。

AWS SDK for Ruby でのオブザーバビリティ機能の設定

オブザーバビリティとは、システムの現在の状態を、それが出力するデータから推測できる範囲です。出力されるデータは、一般的にテレメトリと呼ばれます。AWS SDK for Ruby は、トレースをテレメトリシグナルとして提供できます。をワイヤアップ `TelemetryProvider` してテレメトリデータを収集し、オブザーバビリティバックエンドに送信できます。SDK は現在、テレメトリプロバイダーとして OpenTelemetry (OTel) をサポートしており、OpenTelemetry には、[AWS X-Ray](#) や [Amazon CloudWatch](#) の使用など、テレメトリデータをエクスポートするさまざまな方法があります。Ruby 用の OpenTelemetry エクスポートの詳細については、OpenTelemetry ウェブサイトの「[エクスポーター](#)」を参照してください。

デフォルトでは、SDK はテレメトリデータを記録または出力しません。このトピックでは、テレメトリ出力を設定および出力する方法について説明します。

テレメトリは、特定のサービスまたはグローバルに設定できます。SDK for Ruby は OpenTelemetry プロバイダーを提供します。選択したカスタムテレメトリプロバイダーを定義することもできます。

サービスクライアントの **OTelProvider** の設定

SDK for Ruby は、という OpenTelemetry プロバイダーを提供します [OTelProvider](#)。次の例では、Amazon Simple Storage Service サービスクライアントの OpenTelemetry を使用してテレメトリエクスポートを設定します。この簡単な例では、OpenTelemetry の `OTEL_TRACES_EXPORTER` 環境変数を使用して、コードの実行時にトレースをコンソール出力にエクスポートします。の詳細については `OTEL_TRACES_EXPORTER`、OpenTelemetry ドキュメントの「[エクスポーターの選択](#)」を参照してください。

```
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
```

前のコード例は、サービスクライアントのトレース出力を設定する手順を示しています。

1. OpenTelemetry の依存関係が必要です。
 - a. [opentelemetry-sdk](#) を使用するための `Aws::Telemetry::OTelProvider`。
 - b. [opentelemetry-exporter-otlp](#) テレメトリデータをエクスポートするための。
2. `OpenTelemetry::SDK.configure` を呼び出して、OpenTelemetry SDK を設定デフォルトに設定します。
3. SDK for Ruby の OpenTelemetry プロバイダーを使用して、トレースするサービスクライアントに設定オプションとして `OTelProvider` 渡す のインスタンスを作成します。

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

これらのステップを使用して、そのサービスクライアントで呼び出されるメソッドはトレースデータを送信します。

Amazon S3 の `list_buckets` メソッドの呼び出しから生成されたトレース出力の例を次に示します。

OpenTelemetry トレース出力の例

```
#<struct OpenTelemetry::SDK::Trace::SpanData
  name="Handler.NetHttp",
  kind=:internal,
  status=#<OpenTelemetry::Trace::Status:0x0000000011da17bd8 @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\xe\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\xe\x16\xAC\xE6\x1A\xAC%j\x81\xD8",
trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>

```

前のトレース出力には 2 つのデータスパンがあります。各トレースエントリは、1 つ以上の属性のイベントに関する追加のメタデータを提供します。

OTelProvider すべてのサービスクライアントの の設定

前のセクションで説明したように、特定のサービスクライアントのテレメトリを有効にする代わりに、テレメトリをグローバルに有効にすることもできます。

すべての AWS サービスクライアントのテレメトリデータを生成するには、サービスクライアントを作成する `Aws.config` 前に にテレメトリプロバイダーを設定できます。

```

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

```

この設定では、後で作成されたサービスクライアントは自動的にテレメトリを出力します。 `Aws.config` を使用してグローバル設定を設定する方法の詳細については、「」を参照してください [Aws.config](#)。

カスタムテレメトリプロバイダーの設定

OpenTelemetry をテレメトリプロバイダーとして使用しない場合は、AWS SDK for Ruby がカスタムプロバイダーの実装をサポートします。AWS SDK for Ruby GitHub リポジトリで使用できる [OTelProvider実装](#) を例として使用すると役立つ場合があります。その他のコンテキストについては、AWS SDK for Ruby API リファレンスの [Module: Aws::Telemetry](#) の注意事項を参照してください。

スパン属性

トレースはテレメトリの出力です。トレースは 1 つ以上のスパンで構成されます。スパンには、メソッド呼び出しに適した場合に自動的に含まれる追加のメタデータを含む属性があります。SDK for Ruby でサポートされている属性のリストを次に示します。

- 属性名 - トレースに表示されるデータのラベル付けに使用される名前。
- Type - 値のデータ型。
- 説明 - 値が表す内容の説明。

Attribute Name	Type	Description
###	Boolean	True if the unit of work was unsuccessful. Otherwise, false.
exception.message	String	The exception or error message.
exception.stacktrace	String	A stacktrace as provided by the language runtime if available.
exception.type	String	The type (fully qualified name) of the exception or error.
rpc.system	String	The remote system identifier set to 'aws-api'.

<code>rpc.method</code>	String	The name of the operation being invoked.
<code>rpc.service</code>	String	The name of the remote service.
<code>aws.request_id</code>	String	The AWS request ID returned in the response headers, per HTTP attempt. The latest request ID is used when possible.
<code>code.function</code>	String	The method or function name.
<code>code.namespace</code>	String	The namespace within which <code>code.function</code> is defined.
<code>http.status_code</code>	Long	The HTTP response status code.
<code>http.request_content_length</code>	Long	The size of the request payload body in bytes.
<code>http.response_content_length</code>	Long	The size of the response payload body in bytes.
<code>http.method</code>	String	The HTTP request method.
<code>net.protocol.name</code>	String	The name of the application layer protocol.
<code>net.protocol.version</code>	String	The version of the application layer protocol (e.g. 1.0, 1.1, 2.0).
<code>net.peer.name</code>	String	The logical remote hostname.
<code>net.peer.port</code>	String	The logical remote port number.

 Tip

OpenTelemetry-Ruby には、SDK for Ruby の既存のテレメトリサポートと統合された追加の実装があります。詳細については、open-telemetryGitHub リポジトリの[OpenTelemetry AWS-SDK Instrumentation](#)」を参照してください。

AWS SDK for Ruby 内での HTTP レベルの設定の構成

標準外のエンドポイントを設定する

リージョンは、AWS リクエストに使用する SSL エンドポイントを構築するために使用されます。選択したリージョンで非標準のエンドポイントを使用する必要がある場合は、`Aws.config` に `endpoint` エントリを追加してください。または、サービスクライアントまたはリソースオブジェクトを作成するときに `endpoint:` を設定します。次の例では、`other_endpoint` エンドポイントに Amazon S3 リソースオブジェクトを作成します。

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

API リクエストに任意のエンドポイントを使用し、その選択内容を維持するには、「AWS SDK およびツールリファレンスガイド」の「[サービス固有のエンドポイント](#)」設定オプションを参照してください。

AWS SDK for Ruby を使用する

このセクションでは、AWS SDK for Ruby を使用したソフトウェアの開発について説明します。これには、SDK の高度な機能の一部を使用する方法が含まれます。

[AWS SDKs およびツールリファレンスガイド](#)には、多くの AWS SDKs で一般的な設定、機能、その他の基本的な概念も含まれています。

トピック

- [AWS SDK for Ruby REPL ユーティリティの使用](#)
- [AWS Ruby on Rails での SDK for Ruby の使用](#)
- [AWS SDK for Ruby クライアントからのワイヤトレース情報を使用したデバッグ](#)
- [AWS SDK for Ruby アプリケーションへのスタブによるテストの追加](#)
- [AWS SDK for Ruby でページ分割された結果を使用する](#)
- [AWS SDK for Ruby でのウェーターの使用](#)

AWS SDK for Ruby REPL ユーティリティの使用

aws-sdk gem には、SDK for Ruby をテストして結果をすぐに確認できる、読み取り-評価-印刷-ループ (REPL) のインタラクティブなコマンドラインインターフェイスが用意されています。Ruby Gems 用の SDK は [RubyGems.org](https://rubygems.org) で入手できます。

前提条件

- [AWS SDK for Ruby のインストール](#).
- [aws-v3.rb](#) ファイルは [aws-sdk-resources](#) gem にあります。この aws-sdk-resources gem はメインの [aws-sdk](#) gem にも含まれています。
- rexml gem などの XML ライブラリが必要になります。
- このプログラムは Interactive Ruby Shell (irb)でも動作しますが、より強力な REPL 環境を提供する pry gem のインストールをお勧めします。

バンドラーの設定

[Bundler](#) を使用している場合、以下の Gemfile へのアップデートで、前提条件となる gem に対応できます：

1. AWS SDK for Ruby をインストールしたときに作成した Gemfile を開きます。このファイルに次の行を追加します。

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

2. Gem ファイルを保存します。
3. お使いの Gemfile で指定されている依存関係をインストールします。

```
$ bundle install
```

REPL の実行

コマンドラインから `aws-v3.rb` を実行すると、REPL にアクセスできます。

```
aws-v3.rb
```

また、verbose フラグを設定すれば HTTP ワイヤロギングを有効にできます。HTTP ワイヤログ記録は、AWS SDK for Ruby と 間の通信に関する情報を提供します AWS。verbose フラグでは、オーバーヘッドが増加し、コードの実行速度が低下するおそれもあるので注意してください。

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

SDK for Ruby には、AWS のサービスへのインターフェイスを提供するクライアントクラスが含まれています。各クライアントクラスは特定の をサポートします AWS のサービス。REPL では、すべてのサービスクラスに、そのサービスとやりとりするための新しいクライアントオブジェクトを返すヘルパーがあります。ヘルパーの名前は、小文字に変換されたサービスの名前になります。例えば、Amazon S3 および Amazon EC2 のヘルパーオブジェクトの名前は、それぞれ `s3` と `ec2` です。アカウントの Amazon S3 バケットを一覧表示するには、プロンプトに `s3.list_buckets` を入力します。

REPL プロンプトに `quit` を入力して終了できます。

AWS Ruby on Rails での SDK for Ruby の使用

[Ruby on Rails](#) は、Ruby でウェブサイトを簡単に作成できるウェブ開発フレームワークを提供します。

AWS は、Rails `aws-sdk-rails` との統合を容易にする gem を提供します。AWS Elastic Beanstalk、AWS OpsWorks AWS CodeDeploy、または [AWS Rails プロビジョナー](#) を使用して、AWS クラウドで Rails アプリケーションをデプロイして実行できます。

`aws-sdk-rails` gem のインストールと使用の詳細については、GitHub リポジトリ <https://github.com/aws/aws-sdk-rails> を参照してください。

AWS SDK for Ruby クライアントからのワイヤトレース情報を使用したデバッグ

`http_wire_trace` ブール値を設定することで、AWS クライアントからワイヤトレース情報を取得できます。ワイヤトレース情報により、クライアントの変更、サービスの問題、ユーザーエラーの区別ができます。`true` の場合、設定にはネットワーク上で何が送信されているかが示されます。次の例では、クライアント作成時にワイヤトレーシングを有効にして Amazon S3 クライアントを作成します。

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

次のコードと引数 `bucket_name` が付与された場合、出力はその名前のバケットが存在するかどうかを示すメッセージを表示します。

```
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
```

バケットが存在する場合、出力は以下のような内容になります。(読みやすくするために HEAD 行に返却が追加されました。)

```
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/
iAX5E30XRBzJ37cfc8T4D7ELC1KFELM=\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
```

クライアントの作成後にワイヤトレースを有効にすることもできます。

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

レポートされるワイヤトレース情報のフィールドの詳細については、「[Transfer Family 必須リクエストヘッダー](#)」を参照してください。

AWS SDK for Ruby アプリケーションへのスタブによるテストの追加

AWS SDK for Ruby アプリケーションでクライアントレスポンスとクライアントエラーをスタブする方法について説明します。

クライアントレスポンスをスタブする

レスポンスをスタブすると、AWS SDK for Ruby はネットワークトラフィックを無効にし、クライアントはスタブされた (またはフェイク) データを返します。スタブされたデータを指定しない場合、クライアントは次のように返します。

- リストは空の配列として
- マップは空のハッシュとして
- 数値はゼロで
- 日付は `now` で

次の例では、Amazon S3 バケットのリストのスタブされた名前を返します。

```
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
```

このコードを実行すると、以下が表示されます。

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

Note

スタブされたデータを指定すると、デフォルト値は、残りのインスタンス属性に適用されなくなります。これは、前の例で、残りのインスタンス属性の `creation_date` は、`now` ではなく、`nil` であることを意味します。

AWS SDK for Ruby は、スタブされたデータを検証します。間違った種類のデータを渡すと、`ArgumentError` 例外を生成します。たとえば、`bucket_data` への前の割り当ての代わりに、以下を使用したとします。

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

AWS SDK for Ruby では、2 つの `ArgumentError` 例外が発生します。

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

クライアントエラーをスタブする

AWS SDK for Ruby が特定のメソッドに対して発生するエラーをスタブすることもできます。次の例では `Caught Timeout::Error error calling head_bucket on aws-sdk` が表示されます。

```
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
```

AWS SDK for Ruby でページ分割された結果を使用する

多くの AWS オペレーションは、ペイロードが大きすぎて 1 回のレスポンスで返せない場合に、切り捨てられた結果を返します。代わりに、サービスはデータの一部とトークンを返し、次の項目のセットを取得します。このパターンはページ分割と呼ばれます。

ページ分割レスポンスは列挙可能です

ページ分割レスポンスデータを処理する最も簡単な方法は、次の例のように、レスポンスオブジェクト内で組み込み列挙子を使用することです。

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

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

これにより API コールごとに 1 つの応答オブジェクトが得られ、名前を持つバケットのオブジェクトを列挙されます。SDK はリクエストを完了するために追加のページを取得します。

ページ分割レスポンスを手動で処理

ページングをユーザー自身が処理するには、レスポンスの `next_page?` メソッドを使用して、取得するページがまだあることを確認するか、`last_page?` メソッドを使用して、取得するページがこれ以上ないことを確認します。

ページがまだある場合は、次の例のように、`next_page` を使用して、(? がないことを確認します) 次のページの結果を取得します。

```
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

`next_page` メソッドを呼び出して、取得するページがない場合は、SDK は [Aws::PageableResponse::LastPageError](#) 例外を生成します。

ページ分割データクラス

AWS SDK for Ruby のページ分割データは、ページ分割されたデータへのアクセスを提供するために [Seahorse::Client::Response](#) に含まれている [Aws::PageableResponse](#) クラスによって処理されます。 <https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Seahorse/Client/Response.html>

AWS SDK for Ruby でのウェーターの使用

ウェーターは、クライアントに発生する特定の状態をポーリングするユーティリティメソッドです。ウェーターはサービスクライアントに対して定義されたポーリング間隔で何度か試行した後に

失敗する場合があります。ウェーターの使用法の例については、AWS「コード例リポジトリ」の「Amazon DynamoDB 暗号化クライアントの [create_table](#) メソッド」を参照してください。

ウェーターの呼び出し

ウェーターを呼び出すには、クライアントサービスで `wait_until` を呼び出します。次の例では、ウェーターは、続行する前にインスタンス `i-12345678` が実行するまで待機します。

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

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

最初のパラメータがウェーターの名前で、サービスクライアントに対して固有であり、どのオペレーションを待機しているかを示します。2 番目のパラメータは、ウェーターに呼び出されたクライアントメソッドに渡されるパラメータのハッシュで、ウェーター名に応じて異なります。

待機できるオペレーションの一覧および各オペレーションに必要なクライアントメソッドについては、「`waiter_names`」および、使用しているクライアント用の「`wait_until`」フィールドドキュメントを参照してください。

待機の失敗

待機処理は、以下の例外のいずれかで失敗する場合があります。

[Aws::Writers::Errors::FailureStateError](#)

待機中にエラー状態が発生しました。

[Aws::Writers::Errors::NoSuchWaiterError](#)

指定されたウェーター名は使用しているクライアントに対して定義されていません。

[Aws::Writers::Errors::TooManyAttemptsError](#)

試行数がウェーターの `max_attempts` 値を超えています。

[Aws::Writers::Errors::UnexpectedError](#)

予期しないエラーが待機中に発生しました。

[Aws::Writers::Errors::WaiterFailed](#)

待機中に待機状態の 1 つが超過、または別のエラーが発生しました。

これらのすべてのエラー (NoSuchWaiterError を除く) は、WaiterFailed に基づいています。ウェーターのエラーを見つけるには、次の例に示すように、WaiterFailed を使用します。

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

ウェーターの設定

各ウェーターにはデフォルトのポーリング間隔とプログラムへコントロールを返す前の試行回数の上限があります。これらの値を設定するには、max_attempts および delay: パラメータを wait_until 呼び出しで使います。以下の例では、5 秒間隔でポーリングして、最大 25 秒待機します。

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

待機の失敗を無効にするには、これらのパラメータのいずれかの値を nil に設定します。

ウェーターの拡張

ウェーターの動作を変更するには、各ポーリング試行の前および待機の前にトリガーされたコールバックを登録できます。

次の例は、各試行の待機時間を 2 倍にすることにより、ウェーターのエクスポネンシャルバックオフを実装します。

```
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
```

以下の例では、最大試行回数を無効にし、代わりに、エラーとなるまでの待機時間を 1 時間 (3,600 秒) に設定します。

```
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 code for Ruby コード例

このトピックのコード例は、AWS SDK for Ruby で 使用方法を示しています AWS。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

一部のサービスには、サービス固有のライブラリや関数の活用方法を示す追加のカテゴリ例が含まれています。

サービス

- [SDK for Ruby を使用した Aurora の例](#)
- [SDK for Ruby を使用した Auto Scaling の例](#)
- [SDK for Ruby を使用した CloudTrail の例](#)
- [SDK for Ruby を使用した CloudWatch の例](#)
- [SDK for Ruby を使用した Amazon Cognito ID プロバイダーの例](#)
- [SDK for Ruby を使用した Amazon Comprehend の例](#)
- [SDK for Ruby を使用した Amazon DocumentDB の例](#)
- [SDK for Ruby を使用した DynamoDB の例](#)
- [SDK for Ruby を使用した Amazon EC2 の例](#)
- [SDK for Ruby を使用した Elastic Beanstalk の例](#)
- [SDK for Ruby を使用した EventBridge の例](#)
- [AWS Glue SDK for Ruby を使用した例](#)
- [SDK for Ruby を使用した IAM の例](#)
- [SDK for Ruby を使用した Kinesis の例](#)
- [AWS KMS SDK for Ruby を使用した例](#)

- [SDK for Ruby を使用した Lambda の例](#)
- [SDK for Ruby を使用した Amazon MSK の例](#)
- [SDK for Ruby を使用した Amazon Polly の例](#)
- [SDK for Ruby を使用した Amazon RDS の例](#)
- [SDK for Ruby を使用した Amazon S3 の例](#)
- [SDK for Ruby を使用した Amazon SES の例](#)
- [SDK for Ruby を使用した Amazon SES API v2 の例](#)
- [SDK for Ruby を使用した Amazon SNS の例](#)
- [SDK for Ruby を使用した Amazon SQS の例](#)
- [AWS STS SDK for Ruby を使用した例](#)
- [SDK for Ruby を使用した Amazon Textract の例](#)
- [SDK for Ruby を使用した Amazon Translate の例](#)
- [SDK for Ruby を使用した Amazon WorkDocs の例](#)

SDK for Ruby を使用した Aurora の例

次のコード例は、Aurora AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Aurora

次のコード例は、Aurora の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBClusters](#)」を参照してください。

SDK for Ruby を使用した Auto Scaling の例

次のコード例は、Auto Scaling AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[DescribeAutoScalingGroups](#)」を参照してください。

SDK for Ruby を使用した CloudTrail の例

次のコード例は、CloudTrail AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

CreateTrail

次の例は、CreateTrail を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[CreateTrail](#)を参照してください。

DeleteTrail

次の例は、DeleteTrail を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[DeleteTrail](#)を参照してください。

ListTrails

次の例は、ListTrails を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[ListTrails](#)を参照してください。

LookupEvents

次の例は、LookupEvents を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[LookupEvents](#)を参照してください。

SDK for Ruby を使用した CloudWatch の例

次のコード例は、CloudWatch AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

DescribeAlarms

次の例は、DescribeAlarms を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeAlarms](#)」を参照してください。

DescribeAlarmsForMetric

次の例は、DescribeAlarmsForMetric を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
#
# @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
end

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__

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeAlarmsForMetric](#)」を参照してください。

DisableAlarmActions

次の例は、DisableAlarmActions を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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__
```

- API 詳細については、「AWS SDK for Ruby API リファレンス」の「[DisableAlarmActions](#)」を参照してください。

ListMetrics

次の例は、ListMetrics を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListMetrics](#)」を参照してください。

PutMetricAlarm

次の例は、PutMetricAlarm を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# 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 threshold [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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutMetricAlarm](#)」を参照してください。

PutMetricData

次の例は、PutMetricData を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutMetricData](#)」を参照してください。

SDK for Ruby を使用した Amazon Cognito ID プロバイダーの例

次のコード例は、Amazon Cognito ID プロバイダー AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Amazon Cognito

次のコード例は、Amazon Cognito の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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

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

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListUserPools](#)」を参照してください。

SDK for Ruby を使用した Amazon Comprehend の例

次のコード例は、Amazon Comprehend AWS SDK for Ruby でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [シナリオ](#)

シナリオ

顧客からのフィードバックを分析するアプリケーションの作成

次のコード例は、顧客のコメントカードを分析し、元の言語から翻訳し、顧客の感情を判断し、翻訳されたテキストから音声ファイルを生成するアプリケーションの作成方法を示しています。

SDK for Ruby

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、[GitHub](#) のプロジェクトを参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Ruby を使用した Amazon DocumentDB の例

次のコード例は、Amazon DocumentDB AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [サーバーレスサンプル](#)

サーバーレスサンプル

Amazon DocumentDB トリガーから Lambda 関数を呼び出す

次のコード例は、DocumentDB 変更ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DocumentDB ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で Amazon DocumentDB イベントの消費。

```
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
```

SDK for Ruby を使用した DynamoDB の例

次のコード例は、DynamoDB AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello DynamoDB

次のコード例は、DynamoDB の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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

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

- API の詳細については、『AWS SDK for Ruby API リファレンス』の「[ListTables](#)」を参照してください。

トピック

- [基本](#)
- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- 映画データを保持できるテーブルを作成する。
- テーブルに 1 つの映画を入れ、取得して更新する。
- サンプル JSON ファイルから映画データをテーブルに書き込む。
- 特定の年にリリースされた映画を照会する。
- 何年もの間にリリースされた映画をスキャンする。
- テーブルからムービーを削除し、テーブルを削除します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

DynamoDB テーブルをカプセル化するクラスを作成します。

```
# 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

```

サンプルの JSON ファイルをダウンロードして抽出するヘルパー関数を作成します。

```

# 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
  end
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

```

対話型シナリオを実行してテーブルを作成し、そのテーブルに対してアクションを実行します。

```

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

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

アクション

BatchExecuteStatement

次の例は、BatchExecuteStatement を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

PartiQL を使用して項目のバッチを読み取ります。

```
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
```

PartiQL を使用して項目のバッチを削除します。

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[BatchExecuteStatement](#)」を参照してください。

BatchWriteItem

次の例は、BatchWriteItem を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[BatchWriteItem](#)」を参照してください。

CreateTable

次の例は、CreateTable を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
```

- API の詳細については、『AWS SDK for Ruby API リファレンス』の「[CreateTable](#)」を参照してください。

DeleteItem

次の例は、DeleteItem を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteItem](#)」を参照してください。

DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
  end
end

```

```
@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
```

- APIの詳細については、『AWS SDK for Ruby API リファレンス』の「[DeleteTable](#)」を参照してください。

DescribeTable

次の例は、DescribeTable を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeTable](#)」を参照してください。

ExecuteStatement

次の例は、ExecuteStatement を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

PartiQL を使用して項目を 1 つ選択します。

```

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

```

PartiQL を使用して項目を更新します。

```

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

```

```
end
```

PartiQL を使用して項目を 1 つ追加します。

```
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
end
```

PartiQL を使用して項目を 1 つ削除します。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ExecuteStatement](#)」を参照してください。

GetItem

次の例は、GetItem を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetItem](#)」を参照してください。

ListTables

次の例は、ListTables を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

テーブルが存在するかどうかを確認します。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListTables](#)」を参照してください。

PutItem

次の例は、PutItem を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutItem](#)」を参照してください。

Query

次の例は、Query を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Query](#)」を参照してください。

Scan

次の例は、Scan を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Scan](#)」を参照してください。

UpdateItem

次の例は、UpdateItem を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
  end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateItem](#)」を参照してください。

シナリオ

PartiQL ステートメントのバッチを使用してテーブルにクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- 複数の SELECT ステートメントを実行して、項目のバッチを取得する。
- 複数の INSERT ステートメントを実行して、項目のバッチを追加する。
- 複数の UPDATE ステートメントを実行して、項目のバッチを更新する。
- 複数の DELETE ステートメントを実行して、項目のバッチを削除する。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

テーブルを作成し、PartiQL クエリのバッチを実行するシナリオを実行します。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[BatchExecuteStatement](#)」を参照してください。

PartiQL を使用してテーブルに対してクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- SELECT ステートメントを実行して項目を取得する。
- INSERT 文を実行して項目を追加する。
- UPDATE ステートメントを使用して項目を更新する。
- DELETE ステートメントを実行して項目を削除する。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

テーブルを作成し、PartiQL クエリを実行するシナリオを実行します。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ExecuteStatement](#)」を参照してください。

サーバーレスサンプル

DynamoDB トリガーから Lambda 関数を呼び出す

次のコード例は、DynamoDB ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DynamoDB ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で DynamoDB イベントの消費。

```
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
```

DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で DynamoDB のバッチアイテム失敗のレポート。

```
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
```

SDK for Ruby を使用した Amazon EC2 の例

次のコード例は、Amazon EC2 AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Amazon EC2

以下のコード例は、Amazon EC2 の利用開始方法を表示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeSecurityGroups](#)」を参照してください。

トピック

- [アクション](#)

アクション

AllocateAddress

次の例は、AllocateAddress を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[AllocateAddress](#)」を参照してください。

AssociateAddress

次の例は、AssociateAddress を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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 association 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

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[AssociateAddress](#)」を参照してください。

CreateKeyPair

次の例は、CreateKeyPair を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateKeyPair](#)」を参照してください。

CreateRouteTable

次の例は、CreateRouteTable を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateRouteTable](#)」を参照してください。

CreateSecurityGroup

次の例は、CreateSecurityGroup を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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

```

```

    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)
    end
  end

```

```
    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__

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateSecurityGroup](#)」を参照してください。

CreateSubnet

次の例は、CreateSubnet を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateSubnet](#)」を参照してください。

CreateVpc

次の例は、CreateVpc を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateVpc](#)」を参照してください。

DescribeInstances

次の例は、DescribeInstances を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeInstances](#)」を参照してください。

DescribeRegions

次の例は、DescribeRegions を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.
end
```

```

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

# 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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeRegions](#)」を参照してください。

ReleaseAddress

次の例は、ReleaseAddress を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ReleaseAddress](#)」を参照してください。

StartInstances

次の例は、StartInstances を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[StartInstances](#)」を参照してください。

StopInstances

次の例は、StopInstances を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[StopInstances](#)」を参照してください。

TerminateInstances

次の例は、`TerminateInstances` を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[TerminateInstances](#)」を参照してください。

SDK for Ruby を使用した Elastic Beanstalk の例

次のコード例は、Elastic Beanstalk AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

DescribeApplications

次の例は、DescribeApplications を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end

```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の「[DescribeApplications](#)」を参照してください。

ListAvailableSolutionStacks

次の例は、ListAvailableSolutionStacks を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
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
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[ListAvailableSolutionStacks](#)」を参照してください。

UpdateApplication

次の例は、UpdateApplication を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
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
end
```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[UpdateApplication](#)を参照してください。

SDK for Ruby を使用した EventBridge の例

次のコード例は、EventBridge AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [シナリオ](#)

シナリオ

ルールを作成してトリガーする

次のコード例では、Amazon EventBridge でルールを作成し、トリガーする方法を示します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

関数を正しい順序で呼び出します。

```
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'
```

指定された Amazon Simple Notification Service (Amazon SNS) トピックがこの関数に提供されているトピックの中に存在するかどうかをチェックします。

```
# 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
end
```

```
    false
  end
```

Amazon SNS で発信者が利用できるトピックの中に指定されたトピックが存在するかどうかをチェックします。

```
# 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
```

Amazon SNS でトピックを作成し、E メールアドレスをサブスクライブして、そのトピックに対する通知を受信します。

```
# 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
```

指定された AWS Identity and Access Management (IAM) ロールが、この関数に提供されるロールの中に存在するかどうかを確認します。

```
# 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
```

IAM で発信者が利用できるロールの中に指定されたロールが存在するかどうかをチェックします。

```
# 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
  end
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
puts 'Role not found.'
false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end

```

IAM でロールを作成します。

```

# 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

```

この関数に提供されるルールの中に、指定された EventBridge ルールが存在するかどうかをチェックします。

```
# 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
```

EventBridge で発信者が利用できるルールの中に指定されたルールが存在するかどうかをチェックします。

```
# 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
    puts 'Rule not found.'
    false
  rescue StandardError => e
    puts "Rule not found: #{e.message}"
    false
  end
end

```

Amazon EventBridge でルールを作成します。

```

# 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
end

```

Amazon CloudWatch Logs で発信者が利用できるロググループの中に指定されたロググループが存在するかどうかをチェックします。

```

# 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

```

CloudWatch Logs にロググループを作成します。

```

# 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

```

CloudWatch Logs でログストリームにイベントを書き込みます。

```
# 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

```

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを再起動し、関連するアクティビティに関する情報を CloudWatch Logs 内のログストリームに追加します。

```

# 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

```

EventBridge のルールのアクティビティに関する情報を表示しています。

```

# 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

```

CloudWatch Logs ロググループ内のすべてのログストリームのログ情報を表示しています。

```

# 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
end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end

```

不要になった関連 AWS リソースを手動でクリーンアップするように、発信者にリマインダーを表示します。

```

# 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
```

- APIの詳細については、『AWS SDK for Ruby API リファレンス』の以下のトピックを参照してください。

- [PutEvents](#)
- [PutRule](#)

AWS Glue SDK for Ruby を使用した例

次のコード例は、AWS SDK for Ruby を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Glue。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにちは AWS Glueは

次のコード例は、AWS Glueの使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    end
  end
end
```

```
    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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListJobs](#)」を参照してください。

トピック

- [基本](#)
- [アクション](#)

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- パブリック Amazon S3 バケットをクロールし、CSV 形式のメタデータのデータベースを生成するクローラーを作成する。
- のデータベースとテーブルに関する情報を一覧表示します AWS Glue Data Catalog。
- S3 バケットから CSV 形式のデータを抽出するジョブを作成し、そのデータを変換して JSON 形式の出力を別の S3 バケットにロードする。
- ジョブ実行に関する情報を一覧表示し、変換されたデータを表示してリソースをクリーンアップする。

詳細については、「[チュートリアル: AWS Glue Studio の開始方法](#)」を参照してください。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

シナリオで使用される AWS Glue 関数をラップするクラスを作成します。

```
# 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
```

```
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)
  end
```

```

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

```

シナリオを実行するクラスを作成します。

```

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

```

ジョブの実行中にデータを抽出、変換、ロード AWS Glue するために で使用される ETL スクリプトを作成します。

```

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()
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJob](#)
- [GetJobRuns](#)

- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

アクション

CreateCrawler

次の例は、CreateCrawler を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateCrawler](#)」を参照してください。

CreateJob

次の例は、CreateJob を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateJob](#)」を参照してください。

DeleteCrawler

次の例は、DeleteCrawler を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteCrawler](#)」を参照してください。

DeleteDatabase

次の例は、DeleteDatabase を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteDatabase](#)」を参照してください。

DeleteJob

次の例は、DeleteJob を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteJob](#)」を参照してください。

DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、『AWS SDK for Ruby API リファレンス』の「[DeleteTable](#)」を参照してください。

GetCrawler

次の例は、GetCrawler を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetCrawler](#)」を参照してください。

GetDatabase

次の例は、GetDatabase を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetDatabase](#)」を参照してください。

GetJobRun

次の例は、GetJobRun を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetJobRun](#)」を参照してください。

GetJobRuns

次の例は、GetJobRuns を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetJobRuns](#)」を参照してください。

GetTables

次の例は、GetTables を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetTables](#)」を参照してください。

ListJobs

次の例は、ListJobs を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListJobs](#)」を参照してください。

StartCrawler

次の例は、StartCrawler を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
  end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[StartCrawler](#)」を参照してください。

StartJobRun

次の例は、StartJobRun を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
  end
end
```

```
@logger.error("Glue could not start job run #{name}: \n#{e.message}")
raise
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[StartJobRun](#)」を参照してください。

SDK for Ruby を使用した IAM の例

次のコード例は、IAM AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

IAM へようこそ

次のコード例は、IAM の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListPolicies](#)」を参照してください。

トピック

- [基本](#)
- [アクション](#)

基本

基本を学ぶ

次のコードサンプルは、ユーザーを作成してロールを割り当てる方法を示しています。

Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#) などの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- 指定したアカウントに Amazon S3 バケットへのアクセス権限を付与するロールを作成します。
- ユーザーにロールを引き受けさせるポリシーを追加します。
- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示しリソースをクリーンアップします。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

IAM ユーザーと、Amazon S3 バケットを一覧表示するアクセス権限を付与するロールを作成します。ユーザーには、ロールの引き受けのみ権限があります。ロールを引き受けた後、一時的な認証情報を使用してアカウントのバケットを一覧表示します。

```
# 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
end
```

```
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!')
end

```

```
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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

アクション

AttachRolePolicy

次の例は、AttachRolePolicy を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AttachRolePolicy](#)」を参照してください。

AttachUserPolicy

次の例は、AttachUserPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
  )
end
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[AttachUserPolicy](#)」を参照してください。

CreateAccessKey

次の例は、CreateAccessKey を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# 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
  end
end
```

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateAccessKey](#)」を参照してください。

CreateAccountAlias

次の例は、CreateAccountAlias を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
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
end
```

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateAccountAlias](#)」を参照してください。

CreatePolicy

次の例は、CreatePolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# 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}")
  end
end
```

```
    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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreatePolicy](#)」を参照してください。

CreateRole

次の例は、CreateRole を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateRole](#)」を参照してください。

CreateServiceLinkedRole

次の例は、CreateServiceLinkedRole を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateServiceLinkedRole](#)」を参照してください。

CreateUser

次の例は、CreateUser を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateUser](#)」を参照してください。

DeleteAccessKey

次の例は、DeleteAccessKey を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
  end
```

```
@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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteAccessKey](#)」を参照してください。

DeleteAccountAlias

次の例は、DeleteAccountAlias を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
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}")
  end
end
```

```
    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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteAccountAlias](#)」を参照してください。

DeleteRole

次の例は、DeleteRole を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteRole](#)」を参照してください。

DeleteServerCertificate

次の例は、DeleteServerCertificate を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```

class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end
end

```

```
@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
  )
end
```

```

    @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
end

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteServerCertificate](#)」を参照してください。

DeleteServiceLinkedRole

次の例は、DeleteServiceLinkedRole を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteServiceLinkedRole](#)」を参照してください。

DeleteUser

次の例は、DeleteUser を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteUser](#)」を参照してください。

DeleteUserPolicy

次の例は、DeleteUserPolicy を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteUserPolicy](#)」を参照してください。

DetachRolePolicy

次の例は、DetachRolePolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DetachRolePolicy](#)」を参照してください。

DetachUserPolicy

次の例は、DetachUserPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DetachUserPolicy](#)」を参照してください。

GetAccountPasswordPolicy

次の例は、GetAccountPasswordPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetAccountPasswordPolicy](#)」を参照してください。

GetPolicy

次の例は、GetPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetPolicy](#)」を参照してください。

GetRole

次の例は、GetRole を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetRole](#)」を参照してください。

GetUser

次の例は、GetUser を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[GetUser](#)」を参照してください。

ListAccessKeys

次の例は、ListAccessKeys を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、アクセスキーを一覧表示、作成、非アクティブ化、および削除します。

```
# 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
end
```

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAccessKeys](#)」を参照してください。

ListAccountAliases

次の例は、ListAccountAliases を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

アカウントエイリアスを一覧表示、作成、および削除します。

```
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}")
  end
end
```

```
    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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAccountAliases](#)」を参照してください。

ListAttachedRolePolicies

次の例は、ListAttachedRolePolicies を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# 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
  end
end
```

```

    @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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListAttachedRolePolicies](#)」を参照してください。

ListGroups

次の例は、ListGroups を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListGroup](#)」を参照してください。

ListPolicies

次の例は、ListPolicies を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

このサンプルモジュールは、ロールポリシーを一覧表示、作成、アタッチ、およびデタッチします。

```
# 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
end
```

```
# @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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListPolicies](#)」を参照してください。

ListRolePolicies

次の例は、ListRolePolicies を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListRolePolicies](#)」を参照してください。

ListRoles

次の例は、ListRoles を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListRoles](#)」を参照してください。

ListSAMLProviders

次の例は、ListSAMLProviders を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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:")
  end
end
```

```
@logger.error("\t#{e.code}: #{e.message}")
raise
end
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListSAMLProviders](#)」を参照してください。

ListServerCertificates

次の例は、ListServerCertificates を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```
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
  end
end
```

```
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
```

```
end  
end
```

- API の詳細については、「AWS SDK for Ruby SDK for Rust API リファレンス」の「[ListServerCertificates](#)」を参照してください。

ListUsers

次の例は、ListUsers を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListUsers](#)」を参照してください。

PutUserPolicy

次の例は、PutUserPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutUserPolicy](#)」を参照してください。

UpdateServerCertificate

次の例は、UpdateServerCertificate を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

サーバー証明書を一覧表示、更新、および削除します。

```
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
  end
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateServerCertificate](#)」を参照してください。

UpdateUser

次の例は、UpdateUser を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateUser](#)」を参照してください。

SDK for Ruby を使用した Kinesis の例

次のコード例は、Kinesis AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [サーバーレスサンプル](#)

サーバーレスサンプル

Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での Kinesis イベントの消費。

```
# 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
```

Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で Kinesis バッチアイテム失敗のレポートをします。

```
# 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
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 SDK for Ruby を使用した例

次のコード例は、AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS KMS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

CreateKey

次の例は、CreateKey を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateKey](#)」を参照してください。

Decrypt

次の例は、Decrypt を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Decrypt](#)」を参照してください。

Encrypt

次の例は、Encrypt を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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*')
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Encrypt](#)」を参照してください。

ReEncrypt

次の例は、ReEncrypt を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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*')
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ReEncrypt](#)」を参照してください。

SDK for Ruby を使用した Lambda の例

次のコード例は、Lambda AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Lambda

次のコード例では、Lambda の使用を開始する方法について示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListFunctions](#)」を参照してください。

トピック

- [基本](#)
- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- IAM ロールと Lambda 関数を作成し、ハンドラーコードをアップロードします。
- 1 つのパラメーターで関数を呼び出して、結果を取得します。
- 関数コードを更新し、環境変数で設定します。
- 新しいパラメーターで関数を呼び出して、結果を取得します。返された実行ログを表示します。
- アカウントの関数を一覧表示し、リソースをクリーンアップします。

詳細については、「[コンソールで Lambda 関数を作成する](#)」を参照してください。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ログを書き込むことができる Lambda 関数に前提条件となる IAM アクセス権限を設定します。

```
# 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
end

```

呼び出しパラメータとして指定された数値を増やす Lambda ハンドラーを定義します。

```

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
  end
```

Lambda 関数のデプロイパッケージを圧縮します。

```
# 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
```

新しい Lambda 関数の作成

```
# 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

```

オプションのランタイムパラメータを使用して Lambda 関数を呼び出します。

```

# 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

```

```
end
```

Lambda 関数の設定を更新して、新しい環境変数を挿入します。

```
# 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
end
```

Lambda 関数のコードを、別のコードを含む別のデプロイパッケージで更新します。

```
# 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
```

組み込みのペジネーターを使用して、既存のすべての Lambda 関数を一覧表示します。

```
# 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
```

特定の Lambda 関数を削除します。

```
# 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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

アクション

CreateFunction

次の例は、CreateFunction を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

```

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

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateFunction](#)」を参照してください。

DeleteFunction

次の例は、DeleteFunction を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteFunction](#)」を参照してください。

GetFunction

次の例は、GetFunction を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetFunction](#)」を参照してください。

Invoke

次の例は、Invoke を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Invoke](#)」を参照してください。

ListFunctions

次の例は、ListFunctions を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListFunctions](#)」を参照してください。

UpdateFunctionCode

次の例は、UpdateFunctionCode を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateFunctionCode](#)」を参照してください。

UpdateFunctionConfiguration

次の例は、UpdateFunctionConfiguration を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
  end

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[UpdateFunctionConfiguration](#)」を参照してください。

シナリオ

顧客からのフィードバックを分析するアプリケーションの作成

次のコード例は、顧客のコメントカードを分析し、元の言語から翻訳し、顧客の感情を判断し、翻訳されたテキストから音声ファイルを生成するアプリケーションの作成方法を示しています。

SDK for Ruby

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、[GitHub](#) のプロジェクトを参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

サーバーレスサンプル

Lambda 関数での Amazon RDS データベースへの接続

次のコード例は、RDS データベースに接続する Lambda 関数を実装する方法を示しています。この関数は、シンプルなデータベースリクエストを実行し、結果を返します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda 関数での Amazon RDS データベースへの接続

```
# 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
```

Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での Kinesis イベントの消費。

```
# 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
```

DynamoDB トリガーから Lambda 関数を呼び出す

次のコード例は、DynamoDB ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DynamoDB ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で DynamoDB イベントの消費。

```
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
```

Amazon DocumentDB トリガーから Lambda 関数を呼び出す

次のコード例は、DocumentDB 変更ストリームからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は DocumentDB ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で Amazon DocumentDB イベントの消費。

```
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
```

Amazon MSK トリガーから Lambda 関数を呼び出す

次のコード例は、Amazon MSK クラスターからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は MSK ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での Amazon MSK イベントの消費。

```
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
```

Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda での S3 イベントの消費。

```
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
```

Amazon SNS トリガーから Lambda 関数を呼び出す

次のコード例は、SNS トピックからメッセージを受信することによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SNS イベントの消費。

```
# 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
```

Amazon SQS トリガーから Lambda 関数を呼び出す

次のコード例では、SQS キューからメッセージを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SQS イベントの消費。

```
# 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
```

Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で Kinesis バッチアイテム失敗のレポートをします。

```
# 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
```

```
end
```

DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda で DynamoDB のバッチアイテム失敗のレポート。

```
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
```

Amazon SQS トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、SQS キューからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
# 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
end
```

SDK for Ruby を使用した Amazon MSK の例

次のコード例は、Amazon MSK AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [サーバーレスサンプル](#)

サーバーレスサンプル

Amazon MSK トリガーから Lambda 関数を呼び出す

次のコード例は、Amazon MSK クラスターからレコードを受信することによってトリガーされるイベントを受信する Lambda 関数を実装する方法を示しています。関数は MSK ペイロードを取得し、レコードの内容をログ記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での Amazon MSK イベントの消費。

```
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}"
    end
  end
end
```

```
# Decode base64
msg = Base64.decode64(record['value'])
puts "Message: #{msg}"
end
end
end
```

SDK for Ruby を使用した Amazon Polly の例

次のコード例は、Amazon Polly AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)
- [シナリオ](#)

アクション

DescribeVoices

次の例は、DescribeVoices を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- APIの詳細については、AWS SDK for Ruby API リファレンスの「[DescribeVoices](#)」を参照してください。

ListLexicons

次の例は、ListLexicons を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[ListLexicons](#)」を参照してください。

SynthesizeSpeech

次の例は、SynthesizeSpeech を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[SynthesizeSpeech](#)」を参照してください。

シナリオ

顧客からのフィードバックを分析するアプリケーションの作成

次のコード例は、顧客のコメントカードを分析し、元の言語から翻訳し、顧客の感情を判断し、翻訳されたテキストから音声ファイルを生成するアプリケーションの作成方法を示しています。

SDK for Ruby

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、[GitHub](#) のプロジェクトを参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Ruby を使用した Amazon RDS の例

次のコード例は、Amazon RDS AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Amazon RDS

次のコード例は、Amazon RDS の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBInstances](#)」を参照してください。

トピック

- [アクション](#)
- [サーバーレスサンプル](#)

アクション

CreateDBSnapshot

次の例は、CreateDBSnapshot を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateDBSnapshot](#)」を参照してください。

DescribeDBInstances

次の例は、DescribeDBInstances を使用する方法を説明しています。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBInstances](#)」を参照してください。

DescribeDBParameterGroups

次の例は、DescribeDBParameterGroups を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBParameterGroups](#)」を参照してください。

DescribeDBParameters

次の例は、DescribeDBParameters を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

```

    })

    end

    parameter_groups
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list parameter groups:\n #{e.message}"
  end
end

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBParameters](#)」を参照してください。

DescribeDBSnapshots

次の例は、DescribeDBSnapshots を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

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

```

```
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DescribeDBSnapshots](#)」を参照してください。

サーバーレスサンプル

Lambda 関数での Amazon RDS データベースへの接続

次のコード例は、RDS データベースに接続する Lambda 関数を実装する方法を示しています。この関数は、シンプルなデータベースリクエストを実行し、結果を返します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda 関数での Amazon RDS データベースへの接続

```
# 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
```

SDK for Ruby を使用した Amazon S3 の例

次のコード例は、Amazon S3 AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListBuckets](#)」を参照してください。

トピック

- [基本](#)
- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。

- バケットオブジェクトとバケットを削除します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
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!
  end
end
```

```
        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
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)

- [PutObject](#)

アクション

CopyObject

次の例は、CopyObject を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

オブジェクトをコピーします。

```
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)
  end
end
```

```

    rescue Aws::Errors::ServiceError => e
      puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
#{e.message}"
    end
  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__

```

オブジェクトをコピーし、宛先オブジェクトにサーバー側の暗号化を追加します。

```

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__

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CopyObject](#)」を参照してください。

CreateBucket

次の例は、CreateBucket を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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!'
    end
  end
end
```

```

    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__

```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateBucket](#)」を参照してください。

DeleteBucket

次の例は、DeleteBucket を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteBucket](#)」を参照してください。

DeleteBucketCors

次の例は、DeleteBucketCors を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
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
```

- API の詳細については、「AWS SDK for Ruby SDK for Kotlin API リファレンス」の「[DeleteBucketCors](#)」を参照してください。

DeleteBucketPolicy

次の例は、DeleteBucketPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby SDK for Kotlin API リファレンス」の「[DeleteBucketPolicy](#)」を参照してください。

DeleteObjects

次の例は、DeleteObjects を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteObjects](#)」を参照してください。

GetBucketCors

次の例は、GetBucketCors を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

```
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetBucketCors](#)」を参照してください。

GetBucketPolicy

次の例は、GetBucketPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

```
end
```

- API の詳細については、「AWS SDK for Ruby SDK for Rust API リファレンス」の「[GetBucketPolicy](#)」を参照してください。

GetObject

次の例は、GetObject を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

オブジェクトを取得します。

```
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__
```

オブジェクトを取得し、サーバー側の暗号化状態をレポートします。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetObject](#)」を参照してください。

HeadObject

次の例は、HeadObject を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[HeadObject](#)」を参照してください。

ListBuckets

次の例は、ListBuckets を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ListBuckets](#)」を参照してください。

ListObjectsV2

次の例は、ListObjectsV2 を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListObjectsV2](#)」を参照してください。

PutBucketCors

次の例は、PutBucketCors を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[PutBucketCors](#)」を参照してください。

PutBucketPolicy

次の例は、PutBucketPolicy を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutBucketPolicy](#)」を参照してください。

PutBucketWebsite

次の例は、PutBucketWebsite を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__

```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutBucketWebsite](#)」を参照してください。

PutObject

次の例は、PutObject を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

マネージドアップローダー (Object.upload_file) を使用してファイルをアップロードします。

```
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
```

```
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__
```

Object.put を使用してファイルをアップロードします。

```
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__
```

Object.put を使用してファイルをアップロードし、サーバー側の暗号化を追加します。

```
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__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[PutObject](#)」を参照してください。

シナリオ

署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL を作成し、オブジェクトをアップロードする方法を示しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

サーバーレスサンプル

Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

SDK for Ruby

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用して Lambda での S3 イベントの消費。

```
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
```

SDK for Ruby を使用した Amazon SES の例

次のコード例は、Amazon SES AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

GetIdentityVerificationAttributes

次の例は、GetIdentityVerificationAttributes を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[GetIdentityVerificationAttributes](#)」を参照してください。

ListIdentities

次の例は、ListIdentities を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListIdentities](#)」を参照してください。

SendEmail

次の例は、SendEmail を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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,  
  )  
end
```

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[SendEmail](#)」を参照してください。

VerifyEmailIdentity

次の例は、VerifyEmailIdentity を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}"
end
```

```
# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[VerifyEmailIdentity](#)」を参照してください。

SDK for Ruby を使用した Amazon SES API v2 の例

次のコード例は、Amazon SES API v2 AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

SendEmail

次の例は、SendEmail を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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)
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[SendEmail](#)」を参照してください。

SDK for Ruby を使用した Amazon SNS の例

次のコード例は、Amazon SNS AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)
- [サーバーレスサンプル](#)

アクション

CreateTopic

次の例は、CreateTopic を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、AWS SDK for Ruby デベロッパーガイドを参照してください。
- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateTopic](#)」を参照してください。

ListSubscriptions

次の例は、ListSubscriptions を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「[AWS SDK for Ruby デベロッパーガイド](#)」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「[ListSubscriptions](#)」を参照してください。

ListTopics

次の例は、ListTopics を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
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__
```

- 詳細については、「[AWS SDK for Ruby デベロッパーガイド](#)」を参照してください。

- API の詳細については、AWS SDK for Ruby API リファレンスの「[ListTopics](#)」を参照してください。

Publish

次の例は、Publish を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「[AWS SDK for Ruby デベロッパーガイド](#)」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「[発行](#)」を参照してください。

SetTopicAttributes

次の例は、SetTopicAttributes を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。完全な例を見つけて、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「[AWS SDK for Ruby デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[SetTopicAttributes](#)」を参照してください。

Subscribe

次の例は、Subscribe を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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
end
```

- 詳細については、「[AWS SDK for Ruby デベロッパーガイド](#)」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「[Subscribe](#)」を参照してください。

サーバーレスサンプル

Amazon SNS トリガーから Lambda 関数を呼び出す

次のコード例は、SNS トピックからメッセージを受信することによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SNS イベントの消費。

```
# 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
```

SDK for Ruby を使用した Amazon SQS の例

次のコード例は、Amazon SQS AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)
- [サーバーレスサンプル](#)

アクション

ChangeMessageVisibility

次の例は、ChangeMessageVisibility を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ChangeMessageVisibility](#)」を参照してください。

CreateQueue

次の例は、CreateQueue を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateQueue](#)」を参照してください。

DeleteQueue

次の例は、DeleteQueue を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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)
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[DeleteQueue](#)」を参照してください。

ListQueues

次の例は、ListQueues を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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

```

- API の詳細については、AWS SDK for Ruby API リファレンスの「[ListQueues](#)」を参照してください。

ReceiveMessage

次の例は、ReceiveMessage を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[ReceiveMessage](#)」を参照してください。

SendMessage

次の例は、SendMessage を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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__
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[SendMessage](#)」を参照してください。

SendMessageBatch

次の例は、SendMessageBatch を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
end
```

```
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
```

- APIの詳細については、「AWS SDK for Ruby API リファレンス」の「[SendMessageBatch](#)」を参照してください。

サーバーレスサンプル

Amazon SQS トリガーから Lambda 関数を呼び出す

次のコード例では、SQS キューからメッセージを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SQS イベントの消費。

```
# 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
```

Amazon SQS トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、SQS キューからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
# 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
end
```

AWS STS SDK for Ruby を使用した例

次のコード例は、AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS STS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

AssumeRole

次の例は、AssumeRole を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# 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
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[AssumeRole](#)」を参照してください。

SDK for Ruby を使用した Amazon Textract の例

次のコード例は、Amazon Textract AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [シナリオ](#)

シナリオ

顧客からのフィードバックを分析するアプリケーションの作成

次のコード例は、顧客のコメントカードを分析し、元の言語から翻訳し、顧客の感情を判断し、翻訳されたテキストから音声ファイルを生成するアプリケーションの作成方法を示しています。

SDK for Ruby

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、[GitHub](#) のプロジェクトを参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Ruby を使用した Amazon Translate の例

次のコード例は、Amazon Translate AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [シナリオ](#)

シナリオ

顧客からのフィードバックを分析するアプリケーションの作成

次のコード例は、顧客のコメントカードを分析し、元の言語から翻訳し、顧客の感情を判断し、翻訳されたテキストから音声ファイルを生成するアプリケーションの作成方法を示しています。

SDK for Ruby

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、[GitHub](#) のプロジェクトを参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Ruby を使用した Amazon WorkDocs の例

次のコード例は、Amazon WorkDocs AWS SDK for Ruby で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- [アクション](#)

アクション

DescribeRootFolders

次の例は、DescribeRootFolders を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
# Retrieves the root folder for a user by email
# @param users [Array<Types::User>] A list of users selected from API response
# @param user_email [String] The email of the user.
def get_user_folder(users, user_email)
  user = users.find { |user| user.email_address == user_email }
  if user
    user.root_folder_id
  else
    @logger.error "Could not get root folder for user with email address
#{user_email}"
  end
end
```

```

        exit(1)
      end
    end

    # Describes the contents of a folder
    # @param [String] folder_id - The Id of the folder to describe.
    def describe_folder_contents(folder_id)
      resp = @client.describe_folder_contents({
        folder_id: folder_id, # required
        sort: 'NAME', # accepts DATE, NAME
        order: 'ASCENDING' # accepts
        ASCENDING, DESCENDING
      })

      resp.documents.each do |doc|
        md = doc.latest_version_metadata
        @logger.info "Name:          #{md.name}"
        @logger.info "Size (bytes):  #{md.size}"
        @logger.info "Last modified: #{doc.modified_timestamp}"
        @logger.info "Doc ID:       #{doc.id}"
        @logger.info "Version ID:   #{md.id}"
        @logger.info ''
      end
      rescue Aws::WorkDocs::Errors::ServiceError => e
        @logger.error "Error listing folder contents: #{e.message}"
        exit(1)
      end
    end
  end
end

```

- API の詳細については、AWS SDK for Ruby 「API リファレンス」の[DescribeRootFolders](#)を参照してください。

DescribeUsers

次の例は、DescribeUsers を使用する方法を説明しています。

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

# Describes users within an organization
# @param [String] org_id: The ID of the org.
def describe_users(org_id)
  resp = @client.describe_users({
    organization_id: org_id,
    include: 'ALL', # accepts ALL, ACTIVE_PENDING
    order: 'ASCENDING', # accepts ASCENDING,
DESCENDING
    sort: 'USER_NAME', # accepts USER_NAME
    fields: %w[FULL_NAME STORAGE_LIMIT USER_STATUS
STORAGE_USED] # Corrected field names
  })

  resp.users.each do |user|
    @logger.info "First name:  #{user.given_name}"
    @logger.info "Last name:   #{user.surname}"
    @logger.info "Email:       #{user.email_address}"
    @logger.info "Root folder: #{user.root_folder_id}"
    @logger.info ''
  end
  resp.users
rescue Aws::WorkDocs::Errors::ServiceError => e
  @logger.error "AWS WorkDocs Service Error: #{e.message}"
  exit(1)
end

```

- APIの詳細については、AWS SDK for Ruby「API リファレンス」の「[DescribeUsers](#)」を参照してください。

AWS SDK for Ruby バージョン 1 または 2 から AWS SDK for Ruby バージョン 3 への移行

このトピックには、AWS SDK for Ruby のバージョン 1 または 2 からバージョン 3 への移行に役立つ詳細が含まれています。

サイドバイサイドの使用

AWS SDK for Ruby のバージョン 1 または 2 をバージョン 3 に置き換える必要はありません。同じアプリケーションで併用できます。詳細については、[このブログの投稿](#)を参照してください。

以下に簡単な例を示します。

```
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
```

バージョン 3 の SDK の使用を開始するために、既存の動作中のバージョン 1 または 2 のコードを書き換える必要はありません。有効な移行戦略は、バージョン 3 SDK に対して新しいコードを書くことです。

一般的な違い

バージョン 3 は 1 つの重要な点でバージョン 2 とは異なります。

- 各サービスは個別の Gem として使用できます。

バージョン 2 はいくつかの重要な点でバージョン 1 とは異なります。

- ルート名前空間が異なります。AWS ではなく Aws を使用します。これにより、サイドバイス大度の使用が可能になります。
- Aws.config - メソッドではなくバニラ Ruby ハッシュになりました。

- 厳密なコンストラクタオプション - バージョン 1 SDK でクライアントまたはリソースオブジェクトを構築する場合、不明なコンストラクタは無視されます。バージョン 2 では、不明なコンストラクタオプションは `ArgumentError` をトリガーします。例:

```
# 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
```

クライアントの違い

バージョン 2 とバージョン 3 のクライアントクラスには違いはありません。

バージョン 1 とバージョン 2 の間で、クライアントクラスにはわずかな外部相違点があります。多くのサービスクライアントでは、クライアントの構築後、インターフェイスに互換性があります。重要な相違の一部は以下のとおりです。

- `Aws::S3::Client` - バージョン 1 の Amazon S3 クライアントのクラスはハンドコードでした。バージョン 2 では、サービスモデルから生成されます。メソッドの名前および入力がバージョン 2 では大きく異なります。
- `Aws::EC2::Client` - バージョン 2 では出力リストに複数の名前を使用します。バージョン 1 ではサフィックス `_set` を使用します。例:

```
# 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` - バージョン 2 では構造化レスポンスを使用します。バージョン 1 ではバニラ Ruby ハッシュを使用します。
- サービスクラスの名前変更 - バージョン 2 では、複数のサービスに異なる名前を使用します。

- `AWS::SimpleWorkflow` は `Aws::SWF` になりました
- `AWS::ELB` は `Aws::ElasticLoadBalancing` になりました
- `AWS::SimpleEmailService` は `Aws::SES` になりました
- クライアント設定オプション - バージョン 1 の設定オプションの一部がバージョン 2 で名前変更されました。その他の設定オプションは削除されるか置き換えられています。以下にプライマリの変更を挙げます。
 - `:use_ssl` は削除されました。バージョン 2 ではどこでも SSL を使用します。SSL を無効にするには、`:endpoint` を使用する `http://` を設定する必要があります。
 - `:ssl_ca_file` は `:ssl_ca_bundle` になりました
 - `:ssl_ca_path` は `:ssl_ca_directory` になりました
 - `:ssl_ca_store` が追加されました。
 - `:endpoint` は、ホスト名ではなく完全修飾 HTTP または HTTPS URI にすることが必要になりました。
 - 各サービスの `:_port` オプションが削除され、`:endpoint` に置き換えられました。
 - `:user_agent_prefix` は `:user_agent_suffix` になりました

リソースの違い

バージョン 2 とバージョン 3 のリソースインターフェイスに違いはありません。

バージョン 1 とバージョン 2 では、リソースインターフェイスに大きな違いがあります。バージョン 1 では完全にハンドコードでしたが、バージョン 2 ではリソースインターフェイスはモデルから生成されます。バージョン 2 リソースインターフェイスの方が、はるかに整合性がとれています。システム上の相違点の例を次に示します。

- リソースクラスの分離 - バージョン 2 では、サービス名はモジュールであり、クラスではありません。このモジュールがリソースインターフェイスです。

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- リソースの参照 - バージョン 2 SDK では、リソースの集合と個々のリソースのゲッターが 2 つの異なるメソッドに分離されています。

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- バッチオペレーション - バージョン 1 では、すべてのバッチオペレーションはハンドコードユーティリティでした。バージョン 2 では、多くのバッチオペレーションは API を使用した自動生成バッチオペレーションです。バージョン 2 のバッチインターフェイスは、バージョン 1 と大きく異なります。

AWS SDK for Ruby のセキュリティ

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、AWS とお客様の間で共有される責任です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ – AWS クラウドで提供されているすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。当社のセキュリティ責任は、最優先事項であり AWS、当社のセキュリティの有効性は、[AWS コンプライアンスプログラムの一環として、サードパーティーの監査者によって定期的にテストおよび検証されています](#)。

クラウドにおけるセキュリティ – AWS のサービス お客様の責任は、使用している、およびデータの機密性、組織の要件、適用可能な法律や規制などのその他の要因によって決まります。

トピック

- [AWS SDK for Ruby でのデータ保護](#)
- [AWS SDK for Ruby の Identity and Access Management](#)
- [AWS SDK for Ruby のコンプライアンス検証](#)
- [AWS SDK for Ruby の耐障害性](#)
- [AWS SDK for Ruby のインフラストラクチャセキュリティ](#)
- [AWS SDK for Ruby で最小 TLS バージョンを適用する](#)
- [Amazon S3 暗号化クライアントの移行](#)

AWS SDK for Ruby でのデータ保護

責任 AWS [共有モデル](#)、でのデータ保護に適用されます。このモデルで説明されているように、AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、[データプライバシーに関するよくある質問](#)を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された [AWS 責任共有モデルおよび GDPR](#) のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- を使用して API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の[CloudTrail 証跡の使用](#)を参照してください。
- AWS 暗号化ソリューションと、内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-3](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS CLI または他の AWS のサービス を操作する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

AWS SDK for Ruby の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するのに役立つ Amazon Web Services (AWS) サービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS のサービスリソースの使用を許可する (権限を持たせる) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

AWS SDK for Ruby を使用して にアクセスするには AWS、AWS アカウントと AWS 認証情報が必要です。AWS アカウントのセキュリティを高めるため、アクセス認証情報を提供するには、AWS アカウント認証情報ではなく、IAM ユーザーの利用をお勧めします。

IAM の使用の詳細については、[IAM](#) を参照してください。

IAM ユーザーの概要と、IAM ユーザーがアカウントのセキュリティにとって重要である理由については、[Amazon Web Services 全般的なリファレンス](#)の [AWS セキュリティ認証情報](#)を参照してください。

AWS SDK for Ruby は、サポートする特定の Amazon Web Services (AWS) サービスを通じて[責任共有モデル](#)に従います。AWS のサービス セキュリティ情報については、コンプライアンス[AWS のサービス プログラムによる AWS コンプライアンスの取り組みの対象となるAWS のサービス セキュリティドキュメントページ](#)と を参照してください。

AWS SDK for Ruby のコンプライアンス検証

AWS SDK for Ruby は、サポートする特定の Amazon Web Services (AWS) サービスを通じて[責任共有モデル](#)に従います。AWS のサービス セキュリティ情報については、コンプライアンス[AWS のサービス プログラムによる AWS コンプライアンスの取り組みの対象となるAWS のサービス セキュリティドキュメントページ](#)と を参照してください。

アマゾン ウェブ サービス (AWS) サービスのセキュリティとコンプライアンスは、複数の AWS コンプライアンスプログラムの一環として、サードパーティーの監査者によって評価されます。これには、SOC、PCI、FedRAMP、HIPAA などが含まれます。は、コンプライアンスプログラムによる対象範囲内のサービスで、特定のコンプライアンスプログラム AWS のサービスの範囲内にある の頻繁に更新されるリスト AWS を提供します。 [AWS](#)

サードパーティーの監査レポートは、を使用してダウンロードできます AWS Artifact。詳細については、[AWS 「Artifact でのレポートのダウンロード」](#)を参照してください。

AWS コンプライアンスプログラムの詳細については、[AWS 「コンプライアンスプログラム」](#)を参照してください。

AWS SDK for Ruby を使用して にアクセスする場合のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、組織のコンプライアンス目的、適用可能な法律および規制によって決まります。の使用 AWS のサービス が HIPAA、PCI、FedRAMP などの標準に準拠していることを前提としている場合、 は以下に役立つリソース AWS を提供します。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) – アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順を提供するデプロイガイド AWS。
- [HIPAA 対応サービスのリファレンス](#) – HIPAA 対応サービスの一覧が提供されています。すべて AWS のサービス HIPAA の対象となるわけではありません。

- [AWS コンプライアンスリソース](#) – お客様の業界や地域に適用される可能性のあるワークブックとガイドのコレクション。
- [AWS Config](#) – リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価するサービス。
- [AWS Security Hub](#) – セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立つ AWS 、 内のセキュリティ状態の包括的なビュー。

AWS SDK for Ruby の耐障害性

アマゾン ウェブ サービス (AWS) グローバルインフラストラクチャは、AWS リージョン および アベイラビリティゾーンを中心に構築されています。

AWS リージョン は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティゾーンを提供します。

アベイラビリティゾーンでは、アベイラビリティゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョン およびアベイラビリティゾーンの詳細については、[AWS 「グローバルインフラストラクチャ」](#) を参照してください。

AWS SDK for Ruby は、サポートする特定の Amazon Web Services (AWS) サービスを通じて [責任共有モデル](#) に従います。AWS のサービス セキュリティ情報については、コンプライアンス [AWS のサービス プログラムによる AWS コンプライアンスの取り組みの対象となるAWS のサービス セキュリティドキュメントページ](#) と を参照してください。

AWS SDK for Ruby のインフラストラクチャセキュリティ

AWS SDK for Ruby は、サポートする特定の Amazon Web Services (AWS) サービスを通じて [責任共有モデル](#) に従います。AWS のサービス セキュリティ情報については、コンプライアンス [AWS のサービス プログラムによる AWS コンプライアンスの取り組みの対象となるAWS のサービス セキュリティドキュメントページ](#) と を参照してください。

AWS セキュリティプロセスの詳細については、ホワイトペーパー [AWS 「: セキュリティプロセスの概要」](#) を参照してください。

AWS SDK for Ruby で最小 TLS バージョンを適用する

AWS SDK for Ruby と 間の通信 AWS は、Secure Sockets Layer (SSL) または Transport Layer Security (TLS) を使用して保護されます。SSL のすべてのバージョン、および 1.2 より前の TLS のバージョンには、この通信のセキュリティを侵害する可能性のある脆弱性があります AWS。このため、TLS バージョン 1.2 以降をサポートする Ruby のバージョンで AWS SDK for Ruby を使用していることを確認してください。

Ruby は OpenSSL ライブラリを使用して HTTP 接続を保護します。システム[パッケージマネージャー](#) (yum、apt など)、[公式インストーラー](#)、Ruby [マネージャー](#) (rbenv、RVM、など) でインストールされた、サポートされているバージョンの Ruby (1.9.3 以降) では、通常、TLS 1.2 をサポートする OpenSSL 1.0.1 以降が組み込まれています。

OpenSSL 1.0.1 以降でサポートされているバージョンの Ruby で使用すると、AWS SDK for Ruby は TLS 1.2 を優先し、クライアントとサーバーの両方でサポートされている最新バージョンの SSL または TLS を使用します。これは常に TLS 1.2 以上です AWS のサービス。(SDK は、`use_ssl=true` で `Ruby Net::HTTP` クラスを使用します)。

OpenSSL バージョンの確認

Ruby のインストール環境で OpenSSL 1.0.1 以降が使用されていることを確認するには、次のコマンドを入力します。

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

OpenSSL バージョンを取得する別の方法として、`openssl` 実行可能ファイルに直接クエリを実行できます。最初に、次のコマンドを使用して、適切な実行可能ファイルを検索します。

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

出力には OpenSSL インストール環境の場所を示す `--with-openssl-dir=/path/to/openssl` が含まれています。このパスを書き留めておきます。OpenSSL のバージョンを確認するには、次のコマンドを入力します。

```
cd /path/to/openssl
bin/openssl version
```

この後者の方法は、Ruby の一部のインストール環境では動作しない可能性があります。

TLS サポートのアップグレード

Ruby インストール環境で使用されている OpenSSL のバージョンが 1.0.1 より前の場合は、Ruby の [インストールガイド](#)の説明に従って、システムパッケージマネージャー、Ruby インストーラー、または Ruby マネージャーを使用して Ruby または OpenSSL インストール環境をアップグレードします。Ruby を [ソースから](#)インストールする場合、まず [最新の OpenSSL](#) をインストールしてから、`./configure` の実行時に `--with-openssl-dir=/path/to/upgraded/openssl` を渡します。

Amazon S3 暗号化クライアントの移行

このトピックでは、アプリケーションで使用している Amazon Simple Storage Service (Amazon S3) 暗号化クライアントをバージョン 1 (V1) からバージョン 2 (V2) に移行し、移行プロセス全体でアプリケーションの可用性を確保する方法について説明します。

移行の概要

この移行は 2 つのフェーズから構成されます。

1. 新しいフォーマットを読み取るために既存のクライアントを更新します。まず、AWS SDK for Ruby の更新バージョンをアプリケーションにデプロイします。これにより、既存の V1 暗号化クライアントが、新しい V2 クライアントによって書き込まれたオブジェクトを復号できるようになります。アプリケーションで複数の AWS SDKs、各 SDK を個別にアップグレードする必要があります。
2. 暗号化および復号クライアントを V2 に移行します。すべての V1 暗号化クライアントが新しいフォーマットを読み取ることができるようになったら、既存の暗号化および復号化クライアントをそれぞれの V2 バージョンに移行できます。

新しいフォーマットを読み取るために既存のクライアントを更新する

V2 暗号化クライアントは、古いバージョンのクライアントではサポートされていない暗号化アルゴリズムを使用します。移行の最初のステップとして、V1 復号化クライアントを最新の SDK リリースに更新します。このステップを完了すると、アプリケーションの V1 クライアントが V2 暗号化クライアントによって暗号化されたオブジェクトを復号できるようになります。AWS SDK for Ruby の各メジャーバージョンの詳細については、以下を参照してください。

AWS SDK for Ruby バージョン 3 の更新

バージョン 3 は AWS SDK for Ruby の最新バージョンです。この移行を完了するには、バージョン 1.76.0 以降の `aws-sdk-s3` gem を使用する必要があります。

コマンドラインからインストールする

`aws-sdk-s3 gem` をインストールするプロジェクトの場合、`version` オプションを使用して、1.76.0 以降のバージョンがインストールされていることを確認します。

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Gemfile の使用

Gemfile を使用して依存関係を管理するプロジェクトでは、`aws-sdk-s3 gem` の最小バージョンを 1.76.0 に設定します。例:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Gemfile を変更します。
2. `bundle update aws-sdk-s3` を実行します。バージョンを確認するには、`bundle info aws-sdk-s3` を実行します。

AWS SDK for Ruby バージョン 2 の更新

AWS SDK for Ruby のバージョン 2 は、2021 年 11 月 21 日に[メンテナンスモード](#)に入ります。この移行を完了するには、バージョン 2.11.562 以降の `aws-sdk gem` を使用する必要があります。

コマンドラインからインストールする

`aws-sdk gem` をインストールするプロジェクトの場合、コマンドラインから、`version` オプションを使用して、2.11.562 以降のバージョンがインストールされていることを確認します。

```
gem install aws-sdk -v '>= 2.11.562'
```

Gemfile の使用

Gemfile を使用して依存関係を管理するプロジェクトでは、`aws-sdk gem` の最小バージョンを 2.11.562 に設定します。例:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Gemfile を変更します。Gemfile.lock ファイルがある場合は、それを削除または更新します。

2. `bundle update aws-sdk` を実行します。バージョンを確認するには、`bundle info aws-sdk` を実行します。

暗号化および復号化クライアントを V2 に移行する

クライアントを更新して新しい暗号化形式を読み取るようにした後で、V2 暗号化および復号化クライアントを使用するようにアプリケーションを更新できます。次の手順は、コードを V1 から V2 に正常に移行する方法を示しています。

V2 暗号化クライアントを使用するようにコードを更新する前に、前の手順を実行し、`aws-sdk-s3` gem バージョン 2.11.562 以降を使用していることを確認します。

Note

AES-GCM で復号する場合は、復号されたデータの使用を開始する前に、オブジェクト全体を最後まで読み取ります。これは、オブジェクトが暗号化されてから変更されていないことを確認するためものステップです。

V2 暗号化クライアントを設定する

`EncryptionV2::Client` では追加の設定が必要です。設定の詳細については、[EncryptionV2::Client のドキュメント](#)または、このトピックの後半で示されている例を参照してください。

1. クライアント構成でキーラップ方式とコンテンツ暗号化アルゴリズムを指定する必要があります。新しい `EncryptionV2::Client` を作成する場合、`key_wrap_schema` および `content_encryption_schema` に値を指定する必要があります。

`key_wrap_schema` - を使用している場合は AWS KMS、これを に設定する必要があります:`kms_context`。対称 (AES) キーを使用する場合は、これを `:aes_gcm` に設定する必要があります。非対称 (RSA) キーを使用する場合は、これを `:rsa_oaep_sha1` に設定する必要があります。

`content_encryption_schema` - これは、`:aes_gcm_no_padding` に設定する必要があります。

2. クライアント構成で `security_profile` を指定する必要があります。新しい `EncryptionV2::Client` を作成する場合、`security_profile` に値を指定する必要があります。`security_profile` パラメータは、以前の V1 `Encryption::Client` を使用して書き込まれたオブジェクトの読み取りのサポートを決定します。2 つの値 `:v2` および `:v2_and_legacy` があります。移

行をサポートするには、`security_profile` を `:v2_and_legacy` に設定します。`:v2` は、新しいアプリケーションの開発でのみ使用します。

3. AWS KMS key ID はデフォルトで適用されます。V1 では `Encryption::Client`、クライアントの作成 `kms_key_id` に使用される が に提供されていませんでした AWS KMS Decrypt call。はメタデータからこの情報を取得し、対称暗号文 blob に追加 AWS KMS できます。V2、`EncryptionV2::Client` では、`kms_key_id` が AWS KMS Decrypt 呼び出しに渡され、オブジェクトの暗号化に使用されるキーと一致しない場合、呼び出しは失敗します。コードが以前に特定の `kms_key_id` の設定を行わないことに依存していた場合、クライアント作成時に `kms_key_id: :kms_allow_decrypt_with_any_cmek` を設定するか、`get_object` の呼び出しで `kms_allow_decrypt_with_any_cmek: true` を設定します。

例: 対称 (AES) キーの使用

移行前

```
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)
```

移行後

```
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
```

例: kms_key_id AWS KMS で を使用する

移行前

```
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)
```

移行後

```
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
```

例: kms_key_id AWS KMS を使用せずに を使用する

移行前

```
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)
```

移行後

```
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_cmek:
true) # To allow decrypting with any cmk
```

移行後 (代替方法)

S2 暗号化クライアントを使用してオブジェクトを読み取って復号する場合 (書き込みおよび暗号化は行わない)、以下のコードを使用します。

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: :kms_allow_decrypt_with_any_cmek, # 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
```

ドキュメント履歴

以下の表は、本ガイドの重要な変更点をまとめたものです。このドキュメントの更新に関する通知については、[RSS フィード](#)を購読してください。

変更	説明	日付
コンテンツの再編成	他の AWS SDKs。	2025 年 3 月 29 日
可観測性	Ruby Observability に関する情報を追加しました。	2025 年 1 月 24 日
一般的な更新	Ruby の最小必須バージョンを v2.5 に更新しました。リソースリンクを更新しました。	2024 年 11 月 13 日
目次とガイド付き例	より包括的なコード例リポジトリに延期するガイド付き例を削除しました。	2024 年 7 月 10 日
目次	目次を更新して、コード例をよりわかりやすくしました。	2023 年 6 月 1 日
IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「 IAM のセキュリティのベストプラクティス 」を参照してください。開始方法の更新	2023 年 5 月 8 日
一般的な更新	関連する外部リソースのウェルカムページを更新しました。また、v2.3 に最低限必要な Ruby バージョンも更新しました。用語の更新を反映するように AWS Key Management Service セクションを更新しました。REPL	2022 年 8 月 8 日

ユーティリティの使用情報を
わかりやすく更新しました。

壊れたリンクの修正

壊れたサンプルリンクを修正しました。冗長な「ヒントとコツ」ページを削除し、Amazon EC2 サンプルコンテンツに進むようにしました。コードサンプル リポジトリの GitHub で利用できるコード例のリストを含めました。

2022 年 8 月 3 日

SDK のメトリクス

SDK Metrics for Enterprise Support の有効化に関する情報を削除しました。

2022 年 1 月 28 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。