

开发人员指南

AWS 适用于 Ruby 的 SDK



AWS 适用于 Ruby 的 SDK: 开发人员指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

适用于 Ruby 的 S AWS DK 是什么？	1
其他文档和资源	1
部署到 AWS 云端	1
SDK 主要版本的维护和支持	2
开始使用	3
使用进行身份验证 AWS	3
使用控制台凭证	3
使用 IAM 身份中心身份验证	3
更多身份验证信息	5
安装 SDK	5
先决条件	5
安装 SDK	6
创建简单的应用程序	7
编写代码	7
运行程序	8
Windows 用户的注意事项	8
后续步骤	9
配置服务客户端	10
设置的优先级	11
在外部配置客户端	11
适用于 Ruby 的 AWS SDK 环境变量	12
代码中的客户端配置	12
Aws.config	12
AWS 区域	14
用于解析的区域搜索顺序	14
如何设置区域	14
凭证提供程序	15
凭证提供程序链	16
创建 AWS STS 访问令牌	17
重试	18
在代码中指定客户端重试行为	18
可观测性	19
为服务客户端配置 OTelProvider	19
为所有服务客户端配置 OTelProvider	22

配置自定义遥测提供程序	22
跨度属性	22
HTTP	24
设置非标准端点	24
使用 SDK	25
发出 AWS 服务请求	25
使用 REPL 实用程序	26
先决条件	26
Bundler 设置	26
运行 REPL	27
将开发工具包与 Ruby on Rails 结合使用	27
使用来自客户端的线路跟踪进行调试	27
带模拟的测试	29
模拟客户端响应	29
模拟客户端错误	30
分页	30
分页响应可枚举	31
手动处理分页响应	31
分页数据类	31
Waiter	32
调用 Waiter	32
等待失败	32
配置 Waiter	33
扩展 Waiter	33
代码示例	35
Aurora	36
开始使用	36
Auto Scaling	37
开始使用	36
CloudTrail	39
操作	39
CloudWatch	43
操作	39
Amazon Cognito 身份提供者	56
开始使用	36
Amazon Comprehend	57

场景	58
Amazon DocumentDB	58
无服务器示例	59
DynamoDB	60
开始使用	36
基本功能	62
操作	39
场景	58
无服务器示例	59
Amazon EC2	88
开始使用	36
操作	39
Elastic Beanstalk	122
操作	39
EventBridge	128
场景	58
AWS Glue	146
开始使用	36
基本功能	62
操作	39
IAM	173
开始使用	36
基本功能	62
操作	39
Kinesis	230
无服务器示例	59
AWS KMS	233
操作	39
Lambda	237
开始使用	36
基本功能	62
操作	39
场景	58
无服务器示例	59
Amazon MSK	265
无服务器示例	59

Amazon Polly	266
操作	39
场景	58
Amazon RDS	271
开始使用	36
操作	39
无服务器示例	59
Amazon S3	278
开始使用	36
基本功能	62
操作	39
场景	58
无服务器示例	59
Amazon SES	310
操作	39
Amazon SES API v2	315
操作	39
Amazon SNS	317
操作	39
无服务器示例	59
Amazon SQS	327
操作	39
无服务器示例	59
AWS STS	339
操作	39
Amazon Textract	341
场景	58
Amazon Translate	342
场景	58
迁移版本	344
Side-by-side 用法	344
一般区别	344
客户端区别	345
资源区别	346
安全性	347
数据保护	347

身份和访问管理	348
合规性验证	348
恢复能力	349
基础设施安全性	350
强制实施最低 TLS 版本	350
检查 OpenSSL 版本	350
升级 TLS 支持	351
S3 加密客户端迁移 (从 V1 到 V2)	351
迁移概述	351
更新现有客户端以读取新格式	351
将加密和解密客户端迁移到 V2	353
S3 加密客户端迁移 (V2 到 V3)	356
迁移概述	356
了解 V3 的功能	356
更新现有客户端以读取新格式	359
将加密和解密客户端迁移到 V3	360
文档历史记录	366
.....	ccclxviii

适用于 Ruby 的 AWS SDK 是什么？

欢迎阅读适用于 Ruby 的 AWS SDK 开发人员指南。适用于 Ruby 的 AWS SDK 为几乎所有内容提供支持库 AWS 服务，包括亚马逊简单存储服务 (Amazon S3) Service、亚马逊弹性计算云 (Amazon EC2) Elastic Compute Cloud 和亚马逊 DynamoDB。

AWS 适用于 Ruby 的 SDK 开发者指南提供了有关如何安装、设置和使用适用于 Ruby 的 AWS SDK 来创建使用的 Ruby 应用程序的信息 AWS 服务。

[适用于 Ruby 的 AWS SDK 入门](#)

其他文档和资源

有关适用于 Ruby 的 AWS SDK 开发者的更多资源，请参阅以下内容：

- [AWS SDKs 和工具参考指南](#) — 包含设置、功能和其他常见的基本概念 AWS SDKs
- [适用于 Ruby 的 AWS SDK API 参考-版本 3](#)
- AWS 开启的 [@@ 代码示例存储库](#) GitHub
- [RubyGems.org](#) — 最新版本的 SDK 已模块化为服务专用的 gem，可在此处获得
 - [支持的服务](#)-列出适用于 Ruby 的 AWS SDK 支持的所有宝石
- AWS 适用于 Ruby 的 SDK 来源于 GitHub：
 - [源代码和自述文件](#)
 - [每个 Gem 下的更改日志](#)
 - [从 v2 迁移到 v3](#)
 - [问题](#)
 - [核心升级说明](#)
- [开发人员博客](#)

部署到 AWS 云端

您可以使用 AWS 服务 诸如 AWS Elastic Beanstalk 和之类 AWS CodeDeploy 的将应用程序部署到 AWS 云端。要使用 Elastic Beanstalk 部署 Ruby 应用程序，请参阅[开发者指南中的使用 EB CLI 和 Git 在 Ruby 中部署 EB Beanstalk 应用程序](#)。AWS Elastic Beanstalk 有关 AWS 部署服务的概述，请参阅[AWS上的部署选项概述](#)。

SDK 主要版本的维护和支持

有关 SDK 主要版本及其底层依赖项的维护和支持的信息，请参阅 [《AWS SDKs 和工具参考指南》](#) 中的以下内容：

- [AWS SDKs 和工具维护政策](#)
- [AWS SDKs 和工具版本 Support Matrix](#)

适用于 Ruby 的 AWS SDK 入门

了解如何安装、设置和使用 SDK 创建 Ruby 应用程序，以便以编程方式访问 AWS 资源。

主题

- [AWS 使用 AWS 适用于 Ruby 的 SDK 进行身份验证](#)
- [安装适用于 Ruby 的 S AWS DK](#)
- [使用适用于 Ruby 的 AWS SDK 创建简单应用程序](#)

AWS 使用 AWS 适用于 Ruby 的 SDK 进行身份验证

使用开发 AWS 时，您必须确定您的代码是如何进行身份验证的。AWS 服务您可以根据环境和可用的访问权限以不同的方式配置对 AWS 资源的编程 AWS 访问权限。

要选择您的身份验证方法并针对 SDK 进行配置，请参阅[和工具参考指南中的身份验证AWS SDKs 和访问](#)。

使用控制台凭证

对于本地开发，我们建议新用户使用其现有的 AWS 管理控制台登录凭证以编程方式访问 AWS 服务。基于浏览器的身份验证后，AWS 生成可与 AWS 命令行界面 (CL AWS I) 和 AWS SDK for Ruby 等本地开发工具配合使用的临时证书。

如果您选择此方法，请按照说明使用 [AWS CLI 使用控制台凭据登录进行 AWS 本地开发](#)。

AWS 适用于 Ruby 的 SDK 无需向应用程序中添加额外的 gem (例如aws-sdk-signin) 即可使用控制台凭据登录。

使用 IAM 身份中心身份验证

如果您选择此方法，请完成AWS SDKs 和工具参考指南[中的 IAM Identity Center 身份验证](#)程序。此后，您的环境应包含以下元素：

- AWS CLI，用于在运行应用程序之前启动 AWS 访问门户会话。
- [共享 AWSconfig 文件](#)，其 [default] 配置文件包含一组可从 SDK 中引用的配置值。要查找此文件的位置，请参阅AWS SDKs 和工具参考指南中的[共享文件的位置](#)。
- 共享 config 文件设置了 [region](#) 设置。这将设置 SDK 用于 AWS 请求的默认值 AWS 区域。此区域用于未指定使用区域的 SDK 服务请求。

- 在向 AWS 发送请求之前，SDK 使用配置文件的 [SSO 令牌提供程序配置](#) 来获取凭证。该 `sso_role_name` 值是与 IAM 身份中心权限集关联的 IAM 角色，允许访问您的应用程序中的用户。AWS 服务

以下示例 config 文件展示了使用 SSO 令牌提供程序配置来设置的默认配置文件。配置文件的 `sso_session` 设置是指所指定的 [sso-session 节](#)。该 `sso-session` 部分包含启动 AWS 访问门户会话的设置。

```
[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 适用于 Ruby 的 SDK 无需向您的应用程序添加额外的 gem (例如 `aws-sdk-sso` 和 `aws-sdk-ssooidc`) 即可使用 IAM 身份中心身份验证。

启动 AWS 访问门户会话

在运行可访问的应用程序之前 AWS 服务，您需要为开发工具包进行有效的 AWS 访问门户会话，才能使用 IAM Identity Center 身份验证来解析证书。根据配置的会话时长，访问权限最终将过期，并且开发工具包将遇到身份验证错误。要登录 AWS 访问门户，请在中运行以下命令 AWS CLI。

```
aws sso login
```

如果遵循引导并具有默认的配置文件的设置，则无需使用 `--profile` 选项来调用该命令。如果您的 SSO 令牌提供程序配置在使用指定的配置文件，则命令为 `aws sso login --profile named-profile`。

要选择性测试是否已有活动会话，请运行以下 AWS CLI 命令。

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

如果会话是活动的，则对此命令的响应会报告共享 config 文件中配置的 IAM Identity Center 账户和权限集。

Note

如果您已经有一个有效的 AWS 访问门户会话并且 `aws sso login` 正在运行，则无需提供凭据。

登录过程可能会提示您允许 AWS CLI 访问您的数据。由于 AWS CLI 是在适用于 Python 的 SDK 之上构建的，因此权限消息可能包含 `botocore` 名称的变体。

更多身份验证信息

人类用户，也称为人类身份，是应用程序的人员、管理员、开发人员、操作员和使用者。他们必须具有身份才能访问您的 AWS 环境和应用程序。作为组织成员的人类用户（即您、开发人员）也称为工作人员身份。

访问时使用临时证书 AWS。您可以为人类用户使用身份提供商，通过扮演提供临时证书的角色来提供对 AWS 账户的联合访问权限。对于集中式访问权限管理，我们建议使用 AWS IAM Identity Center (IAM Identity Center) 来管理对您账户的访问权限以及这些账户中的其他权限。有关更多替代方案，请参阅以下内容：

- 有关最佳实践的更多信息，请参阅《IAM 用户指南》中的 [IAM 中的安全最佳实践](#)。
- 要创建短期 AWS 证书，请参阅 IAM 用户指南中的 [临时安全证书](#)。
- 要了解 AWS SDK for Ruby 凭证提供程序链，以及 SDK 如何按顺序自动尝试不同的身份验证方法，请参阅 [凭证提供程序链](#)。
- 有关 AWS SDK 凭据提供程序的配置设置，请参阅《工具参考指南》AWS SDKs 中的 [标准化凭据提供程序](#)。

安装适用于 Ruby 的 AWS SDK

此部分包括针对适用于 Ruby 的 AWS SDK 的先决条件和安装说明。

先决条件

在使用适用于 Ruby 的 AWS SDK 之前，必须使用进行身份验证 AWS。有关设置身份验证的信息，请参阅 [AWS 使用 AWS 适用于 Ruby 的 SDK 进行身份验证](#)。

安装 SDK

你可以像安装任何 Ruby gem 一样安装适用于 Ruby 的 AWS SDK。这些宝石可在以下网址购买[RubyGems](#)。AWS 适用于 Ruby 的 SDK 采用模块化设计，并由以下部分隔开 AWS 服务。整个 aws-sdk Gem 较大，安装过程可能需要一个多小时。

我们建议仅安装供 AWS 服务 您使用的宝石。它们命名为 `likeaws-sdk-service_abbreviation`，完整列表可在 AWS SDK for Ruby 自述文件的“[支持的服务](#)”表中找到。例如，用于与 Amazon S3 服务交互的 Gem 可直接从 [aws-sdk-s3](#) 中获得。

Ruby 版本管理器

我们建议不要使用系统 Ruby，而是使用如下所示的 Ruby 版本管理器：

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

例如，如果您使用的是 Amazon Linux 2 操作系统，则可以使用以下命令更新 RVM，列出可用的 Ruby 版本，然后选择要使用适用于 Ruby 的 AWS SDK 进行开发的版本。所需的最低 Ruby 版本为 2.5。

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

Bundler

如果你使用 [Bundler](#)，则使用以下命令安装适用于 Amazon S3 的 Ruby S AWS DK gem：

1. 安装 Bundler 并创建 Gemfile：

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

2. 打开创建 Gemfile 的，为你的代码将使用的每个 AWS 服务 Gem 添加一行。要按照 Amazon S3 示例进行操作，请将以下行添加到文件底部：

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

3. 保存 Gemfile。
4. 安装 Gemfile 中指定的依赖项：

```
$ bundle install
```

使用适用于 Ruby 的 AWS SDK 创建简单应用程序

使用适用于 Ruby 的 AWS SDK 向亚马逊 S3 打个招呼。以下示例显示了 Amazon S3 存储桶的列表。

编写代码

复制并在新的源文件中粘贴以下代码。将文件命名为 `hello-s3.rb`。

```
require 'aws-sdk-s3'

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

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

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end
```

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

run_demo if $PROGRAM_NAME == __FILE__
```

AWS 适用于 Ruby 的 SDK 采用模块化设计，并采用以下方式分隔 AWS 服务。安装 Gem 后，Ruby 源文件顶部的 `require` 语句会导入 Amazon S3 服务的 AWS SDK 类和方法。有关可用 AWS 服务 gem 的完整列表，请参阅 AWS SDK for Ruby README 文件的“[支持的服务](#)”表。

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

运行程序

打开命令提示符以运行 Ruby 程序。用于运行 Ruby 程序的典型命令语法是：

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

此示例代码不使用任何参数。要运行此代码，请在命令提示符下输入以下内容：

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

Windows 用户的注意事项

在 Windows 上使用 SSL 证书并运行 Ruby 代码时，您可能会看到类似如下的错误。

```
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'
...

```

要修复此问题，请在第一次 AWS 调用之前将以下行添加到 Ruby 源文件中。

```
Aws.use_bundled_cert!
```

如果您在 Ruby 程序中只使用 `aws-sdk-s3` Gem，并且想要使用捆绑证书，则还需要添加 `aws-sdk-core` Gem。

后续步骤

要测试许多其他 Amazon S3 操作，请查看上的 [“AWS 代码示例存储库”](#) GitHub。

在适用于 Ruby 的 AWS SDK 中配置服务客户端

要以编程方式访问 AWS 服务，适用于 Ruby 的 AWS SDK 对每个 AWS 服务使用一个客户端类。例如，如果您的应用程序需要访问 Amazon EC2，则您的应用程序会创建一个 Amazon EC2 客户端对象来与该服务交互。然后，您可以使用服务客户端向该 AWS 服务发出请求。

要向 AWS 服务发出请求，您必须先创建和配置服务客户端。对于您的代码使用的每个 AWS 服务，它都有自己的 gem 和用于与之交互的专用类型。客户端为服务公开的每个 API 操作公开一种方法。

配置 SDK 行为的方法有很多，但归根结底，一切都与服务客户端的行为有关。除非使用基于配置创建的服务客户端，否则任何配置都不会生效。

在使用 AWS 服务进行开发时，您必须确定您的代码是如何使用 AWS 进行身份验证的。您还必须设置要使用的 AWS 区域。

[AWS SDK 和工具参考指南](#)还介绍了在许多 AWS SDK 中常见的设置、功能和其他基础概念。

主题

- [设置的优先级](#)
- [在外部配置适用于 Ruby 的 AWS SDK 服务客户端](#)
- [在代码中为 Ruby 服务客户端配置 AWS SDK](#)
- [为 Ruby AWS 区域的 AWS SDK 进行设置](#)
- [使用 AWS 适用于 Ruby 的 SDK 凭证提供商](#)
- [在适用于 Ruby 的 AWS SDK 中配置重试](#)
- [在 AWS SDK for Ruby 中配置可观察性功能](#)
- [在 Ruby 的 AWS SDK 中配置 HTTP 级别设置](#)

[共享 config 和 credentials 文件](#)可用于配置设置。有关所有 AWS SDK 设置，请参阅《AWS SDK 和工具参考指南》中的[设置参考](#)。

可使用不同的配置文件来存储不同的配置。要指定 SDK 加载的活动配置文件，可以使用 `AWS_PROFILE` 环境变量或 `Aws.config` 的 `profile` 选项。

设置的优先级

全局设置配置了大多数 SDK 支持并在整个 AWS 服务中具有广泛影响的功能、凭证提供者和其他功能。所有 AWS SDK 都有一系列地点（或来源），它们会检查这些地点（或来源），以便找到全局设置的值。并非所有设置在所有来源中都可用。以下是设置查找优先级的方法：

1. 在代码中或服务客户端本身上设置的任何显式设置均优先于其他任何设置。
 - a. 直接传入客户端构造函数的任何参数优先级最高。
 - b. 检查 `Aws.config` 中是否存在全局或服务特定设置。
2. 系统会检查环境变量。
3. 已检查共享 AWS credentials 文件。
4. 已检查共享 AWS config 文件。
5. 最后才会使用适用于 Ruby 的 AWS SDK 源代码本身提供的任何默认值。

在外部配置适用于 Ruby 的 AWS SDK 服务客户端

许多配置设置可以通过代码以外的方式来处理。在外部处理配置时，配置将应用于您的所有应用程序。大多数配置设置可以设置为环境变量，也可以在单独的共享 AWS config 文件中设置。共享 config 文件可以维护多组独立的设置（称为配置文件），以便为不同的环境或测试提供不同的配置。

环境变量和共享 config 文件设置都已标准化，可在 AWS SDK 和工具之间共享，从而支持在不同编程语言和应用程序之间保持一致的功能。

请参阅《AWS SDK 和工具参考指南》，了解如何通过这些方法配置应用程序，以及每个跨 SDK 设置的详细信息。要查看 SDK 可以从环境变量或配置文件中解析的所有设置，请参阅《AWS SDK 和工具参考指南》中的[设置参考](#)。

要向 AWS 服务发出请求，请先实例化该服务对应的客户端。您可以为服务客户端配置常用设置，例如超时、HTTP 客户端及重试配置。

每个服务客户端都需要一个 AWS 区域和一个凭证提供程序。SDK 使用这些值将您的资源请求发送到正确的区域，并使用正确的凭证对请求进行签名。您可以在代码中以编程方式指定这些值，也可以让它们从环境变量中自动加载。

该 SDK 有一系列地点（或来源），它会检查这些地点（或来源），以便找到配置设置的值。

1. 在代码中或服务客户端本身上设置的任何显式设置均优先于其他任何设置。

2. 环境变量

- 有关设置环境变量的详细信息，请参阅《AWS SDK 和工具参考指南》中的[环境变量](#)。
- 请注意，您可以在不同作用域层级为 shell 配置环境变量：系统级、用户级，以及特定终端会话级。

3. 共享config文件和credentials文件

- 有关设置这些文件的详细信息，请参阅《AWS SDK 和工具参考指南》中的[共享 config 和 credentials 文件](#)。

4. 最后才会使用 SDK 源代码本身提供的任何默认值。

- 某些属性（例如“区域”）没有默认值。您必须在代码、环境设置或共享 config 文件中明确指定这些属性。如果 SDK 无法解析所需的配置，API 请求在运行时可能会失败。

适用于 Ruby 的 AWS SDK 环境变量

除了大多数 AWS SDK 支持的[跨 SDK 环境变量](#)外，适用于 Ruby 的 AWS SDK 还支持一些独特的变量：

AWS_SDK_CONFIG_OPT_OUT

如果设置了适用于 Ruby 的 AWS SDK 环境变量 `AWS_SDK_CONFIG_OPT_OUT`，则共享 AWS config 文件（通常位于 `~/.aws/config`）将不会用于任何配置值。

AMAZON_REGION

用于设置 AWS 区域的 `AWS_REGION` 替代环境变量。只有在未使用 `AWS_REGION` 时，才会检查该值。

在代码中为 Ruby 服务客户端配置 AWS SDK

当直接在代码中处理配置时，配置范围仅限于使用该代码的应用程序。在该应用程序中，您可选择以下配置方式：对所有服务客户端的全局配置、对某一特定 AWS 服务类型所有客户端的配置，或对某个具体服务客户端实例的配置。

Aws.config

要在代码中为所有 AWS 类提供全局配置 [Aws.config](#)，请使用 `aws-sdk-core` gem 中提供的全局配置。

`Aws.config` 支持两种不同用途的语法。全局设置可以应用于所有 AWS 服务，也可以应用于特定服务。有关支持的设置的完整列表，请参阅《适用于 Ruby 的 AWS SDK API 参考》中的 [Client Options](#)。

通过 `Aws.config` 进行全局设置

要通过 `Aws.config` 进行与服务无关的设置，请使用以下语法：

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

这些设置将合并到任何已创建的服务客户端中。

全局设置示例：

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

如果您尝试使用不支持全局配置的设置名称，在创建不支持该设置的服务类型实例时将会引发错误。如果发生这种情况，请改用特定于服务的语法。

通过 `Aws.config` 进行特定于服务的设置

要通过 `Aws.config` 进行特定于服务的设置，请使用以下语法：

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

这些设置会合并到该服务类型的所有已创建的服务客户端中。

仅适用于 Amazon S3 的设置示例：

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

可以通过查看对应的[适用于 Ruby 的 AWS SDK gem 名称](#)，并使用紧跟在“aws-sdk-”之后的后缀，来识别 `<service identifier>`。例如：

- 对于 `aws-sdk-s3`，服务标识符字符串为“s3”。
- 对于 `aws-sdk-ecs`，服务标识符字符串为“ecs”。

为 Ruby AWS 区域的 AWS SDK 进行设置

您可以使用访问 AWS 服务 在特定地理区域运营的内容 AWS 区域。它可以用于保证冗余，并保证您的数据和应用程序接近您和用户访问它们的位置。

Important

大多数资源都位于特定的区域 AWS 区域，在使用 SDK 时，您必须为该资源提供正确的区域。

您必须为 Ruby AWS 区域的 SDK 设置默认值才能用于 AWS 请求。此默认设置用于未指定区域的任何 SDK 服务方法调用。

有关该region设置的更多信息，请参阅《AWS SDKs 和工具参考指南》[AWS 区域](#)中的。其中还包括有关如何通过共享 AWS config文件或环境变量设置默认区域的示例。

用于解析的区域搜索顺序

在使用大多数 AWS 服务时，您需要设置区域。AWS 适用于 Ruby 的 SDK 按以下顺序搜索区域：

1. 在客户端或资源对象中设置区域
2. 使用 `Aws.config` 设置区域
3. 使用环境变量设置区域
4. 使用共享config文件设置区域

如何设置区域

此部分从最常见的方法开始介绍设置区域的不同方法。

使用共享 `config` 文件设置区域

通过在共享 AWS config文件中设置region变量来设置区域。有关共享config文件的更多信息，请参阅《工具参考指南》和《工具参考指南》中的[共享配置AWS SDKs 和凭据文件](#)。

在 config 文件中设置此值的示例：

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

如果设置了环境变量 `AWS_SDK_CONFIG_OPT_OUT`，则不会检查共享 `config` 文件。

使用环境变量设置区域

通过设置 `AWS_REGION` 环境变量来设置区域。

使用 `export` 命令在基于 Unix 的系统（例如 Linux 或 macOS）上设置此变量。以下示例将区域设置为 `us-west-2`。

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

要在 Windows 上设置此变量，请使用 `set` 命令。以下示例将区域设置为 `us-west-2`。

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

使用 `Aws.config` 设置区域

通过将 `region` 值添加到 `Aws.config` 哈希中来设置区域。以下示例更新 `Aws.config` 哈希来使用 `us-west-1` 区域。

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

您随后创建的任何客户端或资源都将绑定到此区域。

在客户端或资源对象中设置区域

在创建 AWS 客户机或资源时设置区域。以下示例在 `us-west-1` 区域中创建 Amazon S3 资源对象。为您的 AWS 资源选择正确的区域。服务客户端对象是不可变的，因此您必须为向其发出请求的每个服务创建一个新的客户端，并使用不同的配置向同一服务发出请求。

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

使用 AWS 适用于 Ruby 的 SDK 凭证提供商

对的所有请求都 AWS 必须使用颁发的凭证进行加密签名。AWS 运行时，SDK 会通过检查多个位置来检索凭证的配置值。

身份验证 AWS 可以在您的代码库之外处理。SDK 可通过凭证提供程序链自动检测、使用并刷新多种身份验证方式。

有关项目 AWS 身份验证入门的指导选项，请参阅[和工具参考指南中的身份验证AWS SDKs 和访问权限](#)。

凭证提供程序链

所有 SDKs 人都有一系列地点（或来源）供他们检查，以便获得用于向某人提出请求的有效凭证 AWS 服务。找到有效凭证后，搜索即告停止。这种系统性搜索被称为默认凭证提供程序链。

Note

如果您遵循推荐的方法让新用户开始使用，则在此期间使用控制台凭据登录进行身份验证[AWS 使用 AWS 适用于 Ruby 的 SDK 进行身份验证](#)。其他身份验证方法适用于不同的情况。为避免安全风险，我们建议始终使用短期凭证。有关其他身份验证方法的步骤，请参阅[和工具参考指南中的身份验证AWS SDKs 和访问](#)。

对于链中的每个步骤，都有不同的设置值的方法。直接在代码中设置值始终优先，然后设置为环境变量，然后在共享 AWS config 文件中设置。

《AWS SDKs 和工具参考指南》包含有关所有 AWS SDKs 人使用的 SDK 配置设置的信息 AWS CLI。要详细了解如何通过共享 AWS config 文件配置 SDK，请参阅[共享配置和凭据文件](#)。要详细了解如何通过设置环境变量来配置 SDK，请参阅[环境变量支持](#)。

要进行身份验证 AWS，AWS SDK for Ruby 将按下表所列顺序检查凭证提供者。

凭证提供程序（按优先顺序排列）	AWS SDKs 和工具参考指南	适用于 Ruby 的 AWS SDK API 引用
AWS 访问密钥（临时和长期证书）	AWS 访问密钥	Aws::Credentials Aws::SharedCredentials
来自 AWS Security Token Service (AWS STS) 的 Web 身份令牌	代入角色凭证提供程序 使用 <code>role_arn</code> 、 <code>role_session_name</code> 和 <code>web_identity_token_file</code>	Aws::AssumeRoleWebIdentityCredentials

凭证提供程序 (按优先顺序排列)	AWS SDKs 和工具参考指南	适用于 Ruby 的 AWS SDK API 引用
AWS IAM Identity Center。在本指南中，请参阅 AWS 使用 AWS 适用于 Ruby 的 SDK 进行身份验证 。	IAM Identity Center 凭证提供程序	Aws::SSOCredentials
可信实体提供程序 (如 AWS_ROLE_ARN) 在本指南中，请参阅 创建 AWS STS 访问令牌 。	代入角色凭证提供者 使用 role_arn 和 role_session_name	Aws::AssumeRoleCredentials
登录凭证提供商	登录凭证提供商	Aws::LoginCredentials
流程凭证提供程序	流程凭证提供程序	Aws::ProcessCredentials
Amazon Elastic Container Service (Amazon ECS) 凭证	容器凭证提供程序	Aws::ECSCredentials
亚马逊弹性计算云 (Amazon EC2) 实例配置文件证书 (IMDS 凭证提供商)	IMDS 凭证提供程序	Aws::InstanceProfileCredentials

如果设置了 AWS SDK 的 `AWS_SDK_CONFIG_OPT_OUT` 或 Ruby 环境变量，则不会对共享 AWS config 文件 (通常位于 `~/.aws/config`) 进行凭据解析。

创建 AWS STS 访问令牌

假设角色涉及使用一组临时安全证书，您可以使用这些证书来访问通常可能无法访问的 AWS 资源。这些临时凭证由访问密钥 ID、秘密访问密钥和安全令牌组成。您可以使用 [Aws::AssumeRoleCredentials](#) 方法创建 AWS Security Token Service (AWS STS) 访问令牌。

以下示例使用访问令牌来创建 Amazon S3 客户端对象，其中 `linked::account::arn` 是要代入的角色的 Amazon 资源名称 (ARN)，`session-name` 是代入的角色会话的标识符。

```
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)
```

有关设置 `role_arn` 或 `role_session_name` 改为使用共享 AWS config 文件进行设置的更多信息，请参阅 [AWS SDKs 和工具参考指南](#) 中的代入 [角色凭据提供者](#)。

在适用于 Ruby 的 AWS SDK 中配置重试

适用于 Ruby 的 AWS SDK 为服务请求提供了默认的重试行为和可自定义的配置选项。调用 AWS 服务偶尔会返回意外异常。如果重试调用，某些类型的错误（例如节流或暂时错误）可能会成功。

重试行为可以使用共享 AWS config 文件中的环境变量或设置进行全局配置。有关此方法的信息，请参阅《AWS SDK 和工具参考指南》中的 [重试行为](#)。它还包括有关重试策略实现以及如何选择一种策略而不是另一种策略的详细信息。

或者，您也可以在代码中配置这些选项，如以下部分所示。

在代码中指定客户端重试行为

默认情况下，适用于 Ruby 的 AWS SDK 最多进行三次重试，重试之间间隔 15 秒，总共最多四次尝试。因此，操作可能需要长达 60 秒才超时。

以下示例在区域 `us-west-2` 中创建 Amazon S3 客户端，并指定每个客户端操作的两次重试之间等待五秒。因此，Amazon S3 客户端操作可能需要长达 15 秒才超时。

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

在代码中或服务客户端本身上设置的任何显式设置优先于在环境变量或共享 config 文件中设置的设置。

在 AWS SDK for Ruby 中配置可观察性功能

可观测性是指可以根据系统发出的数据推断出其当前状态的程度。发出的数据通常称为遥测。AWS 适用于 Ruby 的 SDK 可以将轨迹作为遥测信号提供。您可以连接 `TelemetryProvider` 以收集遥测数据并将其发送到可观测性后端。[该软件开发工具包目前支持 OpenTelemetry \(OTel\) 作为遥测提供商，并且 OpenTelemetry 有多种导出遥测数据的方式，包括使用或 AWS X-Ray Amazon CloudWatch 有关 Ruby 导出 OpenTelemetry 出器的更多信息，请参阅 \[OpenTelemetry 网站\]\(#\) 上的导出器。](#)

默认情况下，SDK 不会记录或发出任何遥测数据。本主题介绍了如何配置和发出遥测输出。

遥测既可以针对特定服务进行配置，也可以进行全局配置。适用于 Ruby 的 SDK 提供了一个 `OpenTelemetry` 提供者。您也可以定义自己所选的自定义遥测提供程序。

为服务客户端配置 `OTelProvider`

适用于 Ruby 的 SDK 提供了一个名为的 `OpenTelemetry` 提供程序 `OTelProvider`。以下示例使用 `OpenTelemetry` 亚马逊简单存储服务客户端配置遥测导出。在这个简单的示例中，`OTEL_TRACES_EXPORTER` 环境变量为 `OpenTelemetry` 用于在运行代码时将跟踪导出到控制台输出。要了解更多信息 `OTEL_TRACES_EXPORTER`，请参阅 `OpenTelemetry` 文档中的 [导出器选择](#)。

```
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
```

前面的代码示例显示了为服务客户端配置跟踪输出的步骤：

1. 需要 `OpenTelemetry` 依赖关系。
 - a. [opentelemetry-sdk](#) 用于使用 `Aws::Telemetry::OTelProvider`。
 - b. [opentelemetry-exporter-otlp](#) 用于导出遥测数据。
2. 调用 `OpenTelemetry::SDK.configure` 以使用默认配置设置 `OpenTelemetry` SDK。

3. 使用 SDK for Ruby 的 OpenTelemetry 提供程序，创建OTelProvider要作为配置选项传递给要跟踪的服务客户端的实例。

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

使用这些步骤，在该服务客户端上调用的任何方法都将发出跟踪数据。

通过调用 Amazon S3 的 `list_buckets` 方法生成的跟踪输出示例如下所示：

OpenTelemetry 跟踪输出示例

```
#<struct OpenTelemetry::SDK::Trace::SpanData
  name="Handler.NetHttp",
  kind=:internal,
  status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
  parent_span_id="\xBFb\C9\FD\A6F!\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={}>>

```

之前的跟踪输出有两个数据跨度。每个跟踪条目都通过一个或多个属性提供有关事件的其他元数据。

为所有服务客户端配置 OTelProvider

您可以选择全局开启遥测功能，而不是像上一节所解释的那样为特定的服务客户端开启遥测功能。

要为所有 AWS 服务客户端发送遥测数据，可以在创建服务客户端 `Aws.config` 之前将遥测提供者设置为开启。

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

使用此配置，之后创建的任何服务客户端都将自动发出遥测信号。要了解有关使用 `Aws.config` 设置全局设置的更多信息，请参阅[Aws.config](#)。

配置自定义遥测提供程序

如果您不想 OpenTelemetry 用作遥测提供程序，那么 AWS SDK for Ruby 确实支持您实现自定义提供程序。以 AWS SDK for Ruby GitHub 存储库中提供的[OTelProvider实现](#)作为示例可能会有所帮助。有关其他背景信息，请参阅《适用于 Ruby 的 AWS SDK API 参考》中的 [Module: Aws::Telemetry](#) 的注释。

跨度属性

跟踪是遥测的输出。跟踪由一个或多个跨度组成。跨度包含属性，这些属性会附带额外的元数据；当该元数据适用于方法调用时，会自动纳入其中。以下是适用于 Ruby 的 SDK 所支持的属性列表，其中：

- 属性名称：用于标记跟踪中显示的数据的名称。
- 类型：值的数据类型。
- 描述：值所代表的内容的描述。

属性名称	Type	描述
<code>error</code>	布尔值	如果工作单元不成功，则为 <code>true</code> 。否则为 <code>false</code> 。
<code>exception.message</code>	字符串	异常或错误消息。

<code>exception.stacktrace</code>	字符串	语言运行时提供的堆栈跟踪 (如果有)。
<code>exception.type</code>	字符串	异常或错误的类型 (完全限定名称)。
<code>rpc.system</code>	字符串	远程系统标识符设置为 “aws-api”。
<code>rpc.method</code>	字符串	所调用操作的名称。
<code>rpc.service</code>	字符串	远程服务的名称。
<code>aws.request_id</code>	字符串	每次 HTTP 尝试都会在响应标头中返回的 AWS 请求 ID。尽可能使用最新的请求 ID。
<code>code.function</code>	字符串	方法或函数名称。
<code>code.namespace</code>	字符串	在其中 <code>code.function</code> 定义的命名空间。
<code>http.status_code</code>	长整型	HTTP 响应状态码。
<code>http.request_content_length</code>	长整型	请求负载正文的大小，以字节为单位。
<code>http.response_content_length</code>	长整型	响应负载正文的大小，以字节为单位。
<code>http.method</code>	字符串	HTTP 请求方法。
<code>net.protocol.name</code>	字符串	应用层协议的名称。
<code>net.protocol.version</code>	字符串	应用层协议的版本 (例如 1.0、1.1、2.0)。
<code>net.peer.name</code>	字符串	逻辑远程主机名。
<code>net.peer.port</code>	字符串	逻辑远程端口号。

i Tip

OpenTelemetry-Ruby 还有其他实现与 SDK 集成，适用于 Ruby 现有的遥测支持。有关更多信息，请参阅open-telemetry GitHub 存储库中的 [OpenTelemetry AWS-SDK 工具](#)。

在 Ruby 的 AWS SDK 中配置 HTTP 级别设置

设置非标准端点

该区域用于构造用于 AWS 请求的 SSL 终端节点。如果您需要在所选区域使用非标准端点，请向 `Aws.config` 中添加一个 `endpoint` 条目。或者，在创建服务客户端或资源对象时设置 `endpoint:`。以下示例在 `other_endpoint` 端点中创建 Amazon S3 资源对象。

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

要使用您选择的终端节点来处理 API 请求并使该选择保持不变，请参阅AWS SDKs 和工具参考指南中的 [服务特定端点配置选项](#)。

使用适用于 Ruby 的 AWS SDK

本节提供有关使用 AWS SDK for Ruby 开发软件的信息，包括如何使用 SDK 的某些高级功能。

《[AWS SDKs 和工具参考指南](#)》还包含设置、功能和其他常见的基本概念。AWS SDKs

主题

- [使用适用于 Ruby 的 AWS SDK 发出 AWS 服务请求](#)
- [使用适用于 Ruby 的 AWS SDK REPL 实用程序](#)
- [在 Ruby on Rails 上使用适用于 Ruby 的 AWS SDK](#)
- [使用来自 AWS SDK for Ruby 客户端的线迹信息进行调试](#)
- [在适用于 Ruby 的 AWS SDK 应用程序中添加带模拟的测试](#)
- [在适用于 Ruby 的 AWS SDK 中使用分页结果](#)
- [在 Ruby 的 AWS SDK 中使用服务员](#)

使用适用于 Ruby 的 AWS SDK 发出 AWS 服务请求

要以编程方式访问 AWS 服务，SDK 对每个 AWS 服务使用一个客户端类。例如，如果您的应用程序需要访问 Amazon EC2，则您的应用程序会创建一个 Amazon EC2 客户端对象来与该服务交互。然后，您可以使用服务客户端向该 AWS 服务发出请求。

要向 AWS 服务发出请求，您必须先创建和[配置](#)服务客户端。对于您的代码使用的每个 AWS 服务，它都有自己的 gem 和用于与之交互的专用类型。客户端为服务公开的每个 API 操作公开一种方法。

每个服务客户端都需要一个 AWS 区域和一个凭证提供程序。SDK 使用这些值将您的资源请求发送到正确的区域，并使用正确的凭证对请求进行签名。您可以在代码中以编程方式指定这些值，也可以让它们从环境变量中自动加载。

- 实例化客户端类时，必须提供 AWS 凭证。有关 SDK 检查身份验证提供程序的顺序，请参阅[凭证提供程序链](#)。
- 该 SDK 有一系列地点（或来源），它会检查这些地点（或来源），以便找到配置设置的值。有关更多信息，请参阅[设置的优先级](#)。

适用于 Ruby 的 SDK 包括提供 AWS 服务接口的客户端类。每个客户端类都支持特定的 AWS 服务并遵循 `Aws::<service identifier>::Client` 的命名规范。例如，[Aws::S3::Client](#) 提供与

Amazon Simple Storage Service 服务交互的接口，而 [Aws::SQS::Client](#) 提供与 Amazon Simple Queue Service 服务交互的接口。

所有 AWS 服务的客户端类都是线程安全的。

您可以将配置选项直接传递给客户端和资源构造函数。这些选项的优先级高于环境和 `Aws.config` 默认值。

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

使用适用于 Ruby 的 AWS SDK REPL 实用程序

`aws-sdk` Gem 包含一个读取-求值-输出-循环 (REPL) 交互式命令行界面，您可以在其中测试适用于 Ruby 的 SDK 并立即查看结果。适用于 Ruby 的 SDK Gem 在 [RubyGems.org](#) 中提供。

先决条件

- [安装适用于 Ruby 的 AWS SDK](#).
- [aws-v3.rb](#) 位于 [aws-sdk-resources](#) Gem 中。`aws-sdk-resources` Gem 也包含在主 [aws-sdk](#) Gem 中。
- 您需要一个 xml 库，比如 `rexml` Gem。
- 虽然该程序确实能够使用交互式 Ruby Shell (`irb`)，但我们建议您安装 `pry` Gem，后者提供了更强大的 REPL 环境。

Bundler 设置

如果您使用 [Bundler](#)，则 Gemfile 的以下更新将解决先决条件 Gem 问题：

1. 打开您在安装适用于 Ruby 的 AWS SDK 时创建的 Gemfile。将以下行添加到该文件中：

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

2. 保存此 Gemfile。
3. 安装 Gemfile 中指定的依赖项：

```
$ bundle install
```

运行 REPL

您可以通过从命令行运行 `aws-v3.rb` 来访问 REPL。

```
aws-v3.rb
```

或者，您可以通过设置 `verbose` 标记来启用 HTTP 线路日志记录。HTTP 线路日志记录提供了有关适用于 Ruby 的 AWS SDK 与 AWS 之间通信的信息。请注意，`verbose` 标记还会增加开销，从而使代码运行速度变慢。

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

适用于 Ruby 的 SDK 包括提供 AWS 服务接口的客户端类。每个客户端类都支持特定的 AWS 服务。在 REPL 中，每个服务类都有一个助手，该助手会返回一个用于与该服务交互的新客户端对象。该助手的名称是转换为小写的服务名称。例如，Amazon S3 和 Amazon EC2 助手对象的名称分别为 `s3` 和 `ec2`。要列出您账户中的 Amazon S3 存储桶，您可以在提示中输入 `s3.list_buckets`。

您可以在 REPL 提示中键入 `quit` 以退出。

在 Ruby on Rails 上使用适用于 Ruby 的 AWS SDK

[Ruby on Rails](#) 提供了一个 Web 开发框架，利用它，可轻松地使用 Ruby 创建网站。

AWS 提供了可以轻松与 Rails 集成的 `aws-sdk-rails` 宝石。你可以使用 AWS Elastic Beanstalk、AWS OpsWorks AWS CodeDeploy、或 [AWS Rails Provisioner](#) 在 AWS 云端部署和运行 Rails 应用程序。

有关安装和使用 `aws-sdk-rails` gem 的信息，请参阅 GitHub 存储库 <https://github.com/aws/aws-sdk-rails>。

使用来自 AWS SDK for Ruby 客户端的线迹信息进行调试

通过设置 `http_wire_trace` 布尔值，您可以从 AWS 客户端获取电线追踪信息。线路跟踪信息有助于区分客户端更改、服务问题和用户错误。当值为 `true` 时，该设置会显示线路上当前发送的内容。以下示例创建了 Amazon S3 客户端，该客户端在创建时启用了线路跟踪。

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

鉴于以下代码和参数 `bucket_name`，输出会显示一条消息来表明是否存在该名称的存储桶。

```
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
```

如果存在存储桶，则输出类似于以下内容。(为便于阅读，向 HEAD 行添加了回车。)

```
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
```

您也可以在创建客户端后开启线路跟踪。

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

有关所报告线路跟踪信息中各个字段的更多信息，请参阅 [Transfer Family 必填请求标头](#)。

在适用于 Ruby 的 AWS SDK 应用程序中添加带模拟的测试

了解如何在适用于 Ruby 的 AWS SDK 应用程序中模拟客户端响应和客户端错误。

模拟客户端响应

在模拟响应时，适用于 Ruby 的 AWS SDK 会禁用网络流量，并且客户端会返回模拟的（或伪造的）数据。如果您未提供模拟的数据，则客户端会返回：

- 作为空数组的列表
- 作为空哈希的映射
- 数值 0
- 日期 now

以下示例返回 Amazon S3 存储桶列表的模拟的名称。

```
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
```

运行此代码将显示以下内容。

```
aws-sdk
```

```
aws-sdk2
```

Note

在您提供任何模拟的数据后，默认值将不再应用于任何剩余的实例属性。这意味着在上一个示例中，剩余实例属性 `creation_date` 不是 `now`，而是 `nil`。

适用于 Ruby 的 AWS SDK 会验证模拟的数据。如果您传入的数据的类型错误，则会引发 `ArgumentError` 异常。例如，如果您使用以下内容，而不是 `bucket_data` 以前的分配：

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

适用于 Ruby 的 AWS SDK 将引发两个 `ArgumentError` 异常。

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

模拟客户端错误

您还可以模拟适用于 Ruby 的 AWS SDK 在特定方法中引发的错误。以下示例显示了 `Caught Timeout::Error error calling head_bucket on aws-sdk`。

```
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
```

在适用于 Ruby 的 AWS SDK 中使用分页结果

当有效载荷太大而无法在单个响应中返回时，许多 AWS 操作都会返回截断的结果。相反，该服务会返回一部分数据和一个用于检索下一组项目的令牌。这种模式称为分页。

分页响应可枚举

处理分页响应数据的最简单方法是使用响应对象中的内置枚举器，如以下示例所示。

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

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

这为所进行的每个 API 调用生成一个响应对象，并枚举指定存储桶中的对象。开发工具包检索其他数据页面以完成请求。

手动处理分页响应

要自己处理分页，请使用响应的 `next_page?` 方法来验证是否有其他要检索的更多页面，或使用 `last_page?` 方法来验证是否没有其他要检索的页面。

如果有其他页面，请使用 `next_page` (请注意没有 ?) 方法来检索下一页结果，如以下示例所示。

```
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

如果您调用该 `next_page` 方法，但没有其他页面可供检索，则软件开发工具包会引发 [Aws::PageableResponse::LastPageError](#) 异常。

分页数据类

适用于 Ruby 的 AWS SDK 中的分页数据由 [Aws::](#) 类处理，该 `PageableResponse` 类包含在 [Seahorse::Client::Response](#) 中，用于提供对分页数据的访问权限。

在 Ruby 的 AWS SDK 中使用服务员

Waiter 是实用程序方法，用于轮询客户端上出现的特定状态。按为服务客户端定义的轮询间隔尝试多次后，Waiter 可能会失败。有关如何使用服务员的示例，请参阅代码示例存储库中亚马逊 DynamoDB 加密客户端的 [create_table](#) 方法。AWS

调用 Waiter

要调用 Waiter，请在服务客户端上调用 `wait_until`。在以下示例中，Waiter 一直等到实例 `i-12345678` 运行后才继续。

```
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
```

第一个参数是 Waiter 名称，这特定于服务客户端并指示正在等待哪个操作。第二个参数是传递给 Waiter 所调用的客户端方法的参数的哈希，这因 Waiter 名称而异。

有关可等待的操作和为每个操作调用的客户端方法的列表，请参阅您所使用的客户端的 `waiter_names` 和 `wait_until` 域文档。

等待失败

Waiter 可能会失败并出现以下任何异常。

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

在等待时遇到失败状态。

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

没有为所使用的客户端定义指定的 Waiter 名称。

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

尝试次数超过了 Waiter 的 `max_attempts` 值。

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

在等待时出现意外错误。

[Aws::Writers:: 错误:: WaiterFailed](#)

在等待时超出了等待状态之一或出现了其他故障。

除 `NoSuchWaiterError` 以外的所有其他错误都基于 `WaiterFailed`。要捕获 `Waiter` 中的错误，请使用 `WaiterFailed`，如以下示例所示。

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

配置 Waiter

每个 `Waiter` 都具有默认轮询间隔和它在将控制权返还给您的程序之前将进行尝试的最大次数。要设置这些值，请在您的 `max_attempts` 调用中使用 `delay:` 和 `wait_until` 参数。以下示例最多等待 25 秒，每隔 5 秒轮询一次。

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

要禁用等待失败，请将其中任一参数的值设置为 `nil`。

扩展 Waiter

要修改 `Waiter` 的行为，您可以注册在每个轮询尝试之前和等待之前触发的回调。

以下示例通过使每次尝试后的等待时间长度翻倍，在 `Waiter` 中实现指数回退。

```
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
```

以下示例禁用了最大尝试次数，而在失败前等待 1 小时（3600 秒）。

```
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
```

适用于 Ruby 的 SDK 代码示例

本主题中的代码示例向您展示了如何使用适用于 Ruby 的 AWS SDK 与 AWS。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

某些服务包含其他示例类别，这些类别说明如何利用特定于服务的库或函数。

Services

- [使用适用于 Ruby 的 SDK 的 Aurora 示例](#)
- [使用适用于 Ruby 的 SDK 的自动扩缩示例](#)
- [CloudTrail 使用适用于 Ruby 的 SDK 的示例](#)
- [CloudWatch 使用适用于 Ruby 的 SDK 的示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon Cognito 身份提供者示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon Comprehend 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon DocumentDB 示例](#)
- [使用适用于 Ruby 的 SDK 的 DynamoDB 示例](#)
- [使用适用于 Ruby 的 SDK 的亚马逊 EC2 示例](#)
- [使用适用于 Ruby 的 SDK 的 Elastic Beanstalk 示例](#)
- [EventBridge 使用适用于 Ruby 的 SDK 的示例](#)
- [AWS Glue 使用适用于 Ruby 的 SDK 的示例](#)
- [使用适用于 Ruby 的 SDK 的 IAM 示例](#)
- [使用适用于 Ruby 的 SDK 的 Kinesis 示例](#)
- [AWS KMS 使用适用于 Ruby 的 SDK 的示例](#)
- [使用适用于 Ruby 的 SDK 的 Lambda 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon MSK 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon Polly 示例](#)

- [使用适用于 Ruby 的 SDK 的 Amazon RDS 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon S3 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon SES 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon SES API v2 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon SNS 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon SQS 示例](#)
- [AWS STS 使用适用于 Ruby 的 SDK 的示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon Textract 示例](#)
- [使用适用于 Ruby 的 SDK 的 Amazon Translate 示例](#)

使用适用于 Ruby 的 SDK 的 Aurora 示例

以下代码示例向您展示了如何使用适用于 Ruby 的 AWS SDK 与 Aurora 一起使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)

开始使用

开始使用 Aurora

以下代码示例显示如何开始使用 Aurora。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考DBClusters](#) 中的 [描述](#)。

使用适用于 Ruby 的 SDK 的自动扩缩示例

以下代码示例向您展示了如何使用与 Auto Scaling 适用于 Ruby 的 AWS SDK 配合使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题


- [开始使用](#)

开始使用

开始使用 Auto Scaling

以下代码示例显示了如何开始使用 Auto Scaling。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考 DescribeAutoScalingGroups](#) 中的。

CloudTrail 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 CloudTrail。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateTrail

以下代码示例演示了如何使用 `CreateTrail`。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateTrail](#) 中的。

DeleteTrail

以下代码示例演示了如何使用 DeleteTrail。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteTrail](#) 中的。

ListTrails

以下代码示例演示了如何使用 ListTrails。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListTrails](#) 中的。

LookupEvents

以下代码示例演示了如何使用 LookupEvents。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [LookupEvents](#) 中的。

CloudWatch 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 CloudWatch。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DescribeAlarms

以下代码示例演示了如何使用 DescribeAlarms。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DescribeAlarms](#) 中的。

DescribeAlarmsForMetric

以下代码示例演示了如何使用 DescribeAlarmsForMetric。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
#
# @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
    end
  end
end
```

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

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

    puts 'Dimensions:'
    if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
      alarm.dimensions.each do |d|
        puts "  Name: #{d.name}, Value: #{d.value}"
      end
    else
      puts '  None for this alarm.'
    end
  end
else
  puts 'No alarms found.'
end

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

# Example usage:
def run_me
  region = ''

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

```
else
  region = ARGV[0]
end

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

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DescribeAlarmsForMetric](#)中的。

DisableAlarmActions

以下代码示例演示了如何使用 `DisableAlarmActions`。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DisableAlarmActions](#)中的。

ListMetrics

以下代码示例演示了如何使用 ListMetrics。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListMetrics](#) 中的。

PutMetricAlarm

以下代码示例演示了如何使用 PutMetricAlarm。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

```
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[PutMetricAlarm](#)中的。

PutMetricData

以下代码示例演示了如何使用 PutMetricData。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[PutMetricData](#)中的。

使用适用于 Ruby 的 SDK 的 Amazon Cognito 身份提供者示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Amazon Cognito 身份提供商配合使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)

开始使用

开始使用 Amazon Cognito

以下代码示例显示如何开始使用 Amazon Cognito。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

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

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
```

```
user_pools = []
paginator.each_page do |page|
  user_pools.concat(page.user_pools)
end

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

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListUserPools](#) 中的。

使用适用于 Ruby 的 SDK 的 Amazon Comprehend 示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Amazon Comprehend 配合使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [场景](#)

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

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

使用适用于 Ruby 的 SDK 的 Amazon DocumentDB 示例

以下代码示例向您展示了如何在 Amazon DocumentDB 中 适用于 Ruby 的 AWS SDK 使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [无服务器示例](#)

无服务器示例

通过 Amazon DocumentDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DocumentDB 更改流的记录而触发的事件。该函数检索 DocumentDB 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Ruby 将 Amazon DocumentDB 事件与 Lambda 结合使用。

```
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
```

使用适用于 Ruby 的 SDK 的 DynamoDB 示例

以下代码示例向您展示了如何在 DynamoDB 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [基本功能](#)
- [操作](#)
- [场景](#)
- [无服务器示例](#)

开始使用

开始使用 DynamoDB

以下代码示例显示如何开始使用 DynamoDB。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-dynamodb'
require 'logger'

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListTables](#) 中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建可保存电影数据的表。
- 在表中加入单一电影，获取并更新此电影。
- 向 JSON 示例文件的表中写入电影数据。
- 查询在给定年份发行的电影。
- 扫描在年份范围内发行的电影。
- 删除表中的电影后再删除表。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建封装 DynamoDB 表的类。

```
# 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' }
    ]
  )
end
```

```

    ],
    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

```

创建帮助函数以下载并提取示例 JSON 文件。

```

# Gets sample movie data, either from a local file or by first downloading it from
# the Amazon DynamoDB Developer Guide.
#
# @param movie_file_name [String] The local file name where the movie data is
# stored in JSON format.
# @return [Hash] The movie data as a Hash.
def fetch_movie_data(movie_file_name)
  if !File.file?(movie_file_name)
    @logger.debug("Downloading #{movie_file_name}...")
    movie_content = URI.open(
      'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip'
    )
    movie_json = ''
    Zip::File.open_buffer(movie_content) do |zip|
      zip.each do |entry|
        movie_json = entry.get_input_stream.read
      end
    end
  else
    movie_json = File.read(movie_file_name)
  end
  movie_data = JSON.parse(movie_json)
  # The sample file lists over 4000 movies. This returns only the first 250.
  movie_data.slice(0, 250)
rescue StandardError => e
  puts("Failure downloading movie data:\n#{e}")
  raise
end

```

运行交互式场景以创建表并对其执行操作。

```
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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API 参考》中的以下主题。
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

操作

BatchExecuteStatement

以下代码示例演示了如何使用 BatchExecuteStatement。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 PartiQL 读取一批项目。

```
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
end
```

使用 PartiQL 删除一批项目。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [BatchExecuteStatement](#) 中的。

BatchWriteItem

以下代码示例演示了如何使用 BatchWriteItem。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [BatchWriteItem](#) 中的。

CreateTable

以下代码示例演示了如何使用 CreateTable。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
  end
end
```

```
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[CreateTable](#)中的。

DeleteItem

以下代码示例演示了如何使用 DeleteItem。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteItem](#) 中的。

DeleteTable

以下代码示例演示了如何使用 DeleteTable。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

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

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteTable](#) 中的。

DescribeTable

以下代码示例演示了如何使用 DescribeTable。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DescribeTable](#) 中的。

ExecuteStatement

以下代码示例演示了如何使用 ExecuteStatement。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用 PartiQL 选择单个项目。

```
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
end
```

使用 PartiQL 更新单个项目。

```

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

```

使用 PartiQL 添加单个项目。

```

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
```

使用 PartiQL 删除单个项目。

```
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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ExecuteStatement](#)中的。

GetItem

以下代码示例演示了如何使用 GetItem。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetItem](#) 中的。

ListTables

以下代码示例演示了如何使用 ListTables。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

确定表是否存在。

```
# 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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [ListTables](#) 中的。

PutItem

以下代码示例演示了如何使用 PutItem。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [PutItem](#) 中的。

Query

以下代码示例演示了如何使用 Query。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: { ':year' => year }
    )
    rescue Aws::DynamoDB::Errors::ServiceError => e
      puts("Couldn't query for movies released in #{year}. Here's why:")
      puts("\t#{e.code}: #{e.message}")
      raise
    else
      response.items
    end
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Query](#)。

Scan

以下代码示例演示了如何使用 Scan。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
  }
  done = false
  start_key = nil
  until done
    scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
    response = @table.scan(scan_hash)
    movies.concat(response.items) unless response.items.empty?
    start_key = response.last_evaluated_key
  end
end
```

```
        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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Scan](#)。

UpdateItem

以下代码示例演示了如何使用 UpdateItem。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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] },
    )
  end
end
```

```
        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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [UpdateItem](#) 中的。

场景

使用批量 PartiQL 语句查询表

以下代码示例展示了如何：

- 通过运行多个 SELECT 语句来获取一批项目。
- 通过运行多个 INSERT 语句来添加一批项目。
- 通过运行多个 UPDATE 语句来更新一批项目。
- 通过运行多个 DELETE 语句来删除一批项目。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行创建表并运行批量 PartiQL 查询的场景。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[BatchExecuteStatement](#)中的。

使用 PartiQL 来查询表

以下代码示例展示了如何：

- 通过运行 SELECT 语句来获取项目。

- 通过运行 INSERT 语句来添加项目。
- 通过运行 UPDATE 语句来更新项目。
- 通过运行 DELETE 语句来删除项目。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

运行创建表并运行 PartiQL 查询的场景。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ExecuteStatement](#)中的。

无服务器示例

通过 DynamoDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DynamoDB 流的记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 DynamoDB 事件与 Lambda 结合使用。

```
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
```

通过 DynamoDB 触发器报告 Lambda 函数批处理项目失败

以下代码示例演示如何为接收来自 DynamoDB 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Ruby 通过 Lambda 进行 DynamoDB 批处理项目失败。

```
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
    end
  end
end
```

```
        return {"batchItemFailures" => [{"itemIdentifier" =>
cur_record_sequence_number}]}
      end
    end

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

使用适用于 Ruby 的 SDK 的亚马逊 EC2 示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 EC2。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [操作](#)

开始使用

你好 Amazon EC2

以下代码示例显示了如何开始使用 Amazon EC2。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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.
end
```

```
#
# @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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DescribeSecurityGroups](#) 中的。

操作

AllocateAddress

以下代码示例演示了如何使用 AllocateAddress。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [AllocateAddress](#) 中的。

AssociateAddress

以下代码示例演示了如何使用 AssociateAddress。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[AssociateAddress](#)中的。

CreateKeyPair

以下代码示例演示了如何使用 CreateKeyPair。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

```
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateKeyPair](#) 中的。

CreateRouteTable

以下代码示例演示了如何使用 CreateRouteTable。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateRouteTable](#) 中的。

CreateSecurityGroup

以下代码示例演示了如何使用 CreateSecurityGroup。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# This code example does the following:  
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.  
# 2. Adds inbound rules to the security group.  
# 3. Displays information about available security groups.  
# 4. Deletes the security group.  
  
require 'aws-sdk-ec2'  
  
# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.  
#  
# Prerequisites:  
#  
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).  
#
```

```
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

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

```
# 'sg-030a858e078f1b9EX',
# 'tcp',
# '80',
# '80',
# '0.0.0.0/0'
# )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

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

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
```

```

def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

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

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

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

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

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

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

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

def display_group_tags(tags)

```

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

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

  return if sg.ip_permissions_egress.empty?

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

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

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

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

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

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

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

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

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

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

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
end
```

```

    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__

```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateSecurityGroup](#) 中的。

CreateSubnet

以下代码示例演示了如何使用 CreateSubnet。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateSubnet](#) 中的。

CreateVpc

以下代码示例演示了如何使用 CreateVpc。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateVpc](#) 中的。

DescribeInstances

以下代码示例演示了如何使用 DescribeInstances。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DescribeInstances](#) 中的。

DescribeRegions

以下代码示例演示了如何使用 DescribeRegions。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
# list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print " Endpoint\n"
```

```
print '-' * max_region_string_length
print ' '
print '-' * max_endpoint_string_length
print "\n"
# Print Regions and their endpoints.
result.regions.each do |region|
  print region.region_name
  print ' ' * (max_region_string_length - region.region_name.length)
  print ' '
  print region.endpoint
  print "\n"
end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print ' Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print " State\n"
  print '-' * max_region_string_length
  print ' '
  print '-' * max_zone_string_length
  print ' '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print ' '
```

```
print zone.zone_name
print ' ' * (max_zone_string_length - zone.zone_name.length)
print ' '
print zone.state
# Print any messages for this Availability Zone.
if zone.messages.count.positive?
  print "\n"
  puts ' Messages for this zone:'
  zone.messages.each do |message|
    print "    #{message.message}\n"
  end
end
print "\n"
end
end

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DescribeRegions](#) 中的。

ReleaseAddress

以下代码示例演示了如何使用 ReleaseAddress。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ReleaseAddress](#)中的。

StartInstances

以下代码示例演示了如何使用 StartInstances。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[StartInstances](#)中的。

StopInstances

以下代码示例演示了如何使用 StopInstances。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-ec2'

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

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
```

```
case state
when 'stopping'
  puts 'The instance is already stopping.'
  return true
when 'stopped'
  puts 'The instance is already stopped.'
  return true
when 'terminated'
  puts 'Error stopping instance: ' \
    'the instance is terminated, so you cannot stop it.'
  return false
end
end

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

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

```
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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[StopInstances](#)中的。

TerminateInstances

以下代码示例演示了如何使用 TerminateInstances。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[TerminateInstances](#)中的。

使用适用于 Ruby 的 SDK 的 Elastic Beanstalk 示例

以下代码示例向您展示了如何在 Elastic Beanstalk 中 适用于 Ruby 的 AWS SDK 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DescribeApplications

以下代码示例演示了如何使用 DescribeApplications。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DescribeApplications](#) 中的。

ListAvailableSolutionStacks

以下代码示例演示了如何使用 ListAvailableSolutionStacks。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

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

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

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

  private
end
```

```
# 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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [ListAvailableSolutionStacks](#) 中的。

UpdateApplication

以下代码示例演示了如何使用 UpdateApplication。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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)
  end
end
```

```
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

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

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

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

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

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
```

```
@eb_client.create_application_version(  
  process: false,  
  application_name: @app_name,  
  version_label: version_label,  
  source_bundle: {  
    s3_bucket: fetch_bucket_name,  
    s3_key: "#{@app_name}/#{zip_file_name}"  
  },  
  description: "Updated #{Time.now.strftime('%d/%m/%Y')}}"  
)  
update_environment(version_label)  
rescue Aws::Errors::ServiceError => e  
  @logger.error("Failed to create or deploy application version: #{e.message}")  
end  
  
# Updates the environment to the new application version  
def update_environment(version_label)  
  env_name = fetch_environment_name  
  @eb_client.update_environment(  
    environment_name: env_name,  
    version_label: version_label  
  )  
rescue Aws::Errors::ServiceError => e  
  @logger.error("Failed to update environment: #{e.message}")  
end  
  
# Fetches the environment name of the application  
def fetch_environment_name  
  envs = @eb_client.describe_environments(application_name: @app_name)  
  envs.environments.first.environment_name  
rescue Aws::Errors::ServiceError => e  
  @logger.error("Failed to fetch environment name: #{e.message}")  
  raise  
end  
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [UpdateApplication](#) 中的。

EventBridge 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 EventBridge。适用于 Ruby 的 AWS SDK

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [场景](#)

场景

创建并触发规则

以下代码示例展示了如何在 Amazon 中创建和触发规则 EventBridge。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

按照正确的顺序调用函数。

```
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'
```

检查为该函数提供的主题中是否存在指定的 Amazon Simple Notification Service (Amazon SNS) 主题。

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

检查 Amazon SNS 中可供调用方使用的主题中，是否存在指定主题。

```
# 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

```

在 Amazon SNS 中创建主题，然后订阅一个电子邮件地址，以接收有关该主题的通知。

```

# 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

```

检查为该函数提供的角色中是否存在指定的 AWS Identity and Access Management (IAM) 角色。

```

# 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

```

检查 IAM 中可供调用方使用的角色中，是否存在指定角色。

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

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

在 IAM 中创建一个角色。

```
# 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

```

检查为该函数提供的 EventBridge 规则中是否存在指定的规则。

```

# 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

```

检查指定规则是否存在于中调用者可用的规则中 EventBridge。

```

# Checks whether the specified rule exists among those available to the

```

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

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

在中创建规则 EventBridge。

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
```

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

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

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

在 Amazon Logs 中查看调用者可用的日志组中是否存在指定的 CloudWatch 日志组。

```
# 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
```

在“日志”中创建 CloudWatch 日志组。

```
# 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

```

在 Logs 中将事件写入 CloudWatch 日志流。

```

# 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
```

重启亚马逊弹性计算云 (Amazon EC2) 实例，并将有关相关活动的信息添加到日志中的 CloudWatch 日志流中。

```
# 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

```

显示有关规则活动的信息 EventBridge。

```

# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',

```

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

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

显示日志组中所有日志流的 CloudWatch 日志信息。

```
# 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
```

提醒来电者手动清理他们不再需要的任何关联 AWS 资源。

```
# 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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的以下主题。
 - [PutEvents](#)
 - [PutRule](#)

AWS Glue 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Glue。适用于 Ruby 的 AWS SDK

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [基本功能](#)
- [操作](#)

开始使用

你好 AWS Glue

以下代码示例展示了如何开始使用 AWS Glue。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
```

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

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

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

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

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

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListJobs](#) 中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建爬网程序，爬取公有 Amazon S3 存储桶并生成包含 CSV 格式的元数据的数据库。

- 列出有关数据库和表的信息 AWS Glue Data Catalog。
- 创建任务，从 S3 存储桶提取 CSV 数据，转换数据，然后将 JSON 格式的输出加载到另一个 S3 存储桶中。
- 列出有关作业运行的信息，查看转换后的数据，并清除资源。

有关更多信息，请参阅[教程：AWS Glue Studio 入门](#)。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建一个封装场景中使用的 AWS Glue 函数的类。

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

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

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

# Creates a new crawler with the specified configuration.
#
# @param name [String] The name of the crawler.
# @param role_arn [String] The ARN of the IAM role to be used by the crawler.
# @param db_name [String] The name of the database where the crawler stores its
metadata.
# @param db_prefix [String] The prefix to be added to the names of tables that the
crawler creates.
# @param s3_target [String] The S3 path that the crawler will crawl.
# @return [void]
def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
  @glue_client.create_crawler(
    name: name,
    role: role_arn,
    database_name: db_name,
    targets: {
      s3_targets: [
        {
          path: s3_target
        }
      ]
    }
  )
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not create crawler: \n#{e.message}")
  raise
end

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

# Deletes a crawler with the specified name.
```

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

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

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

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

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

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

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

```
    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
end
```

```
end

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

# Uploads a job script file to an S3 bucket.
#
# @param file_path [String] The local path of the job script file.
# @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
# @return [void]
def upload_job_script(file_path, bucket_resource)
  File.open(file_path) do |file|
    bucket_resource.client.put_object({
      body: file,
      bucket: bucket_resource.name,
      key: file_path
    })
  end
rescue Aws::S3::Errors::S3UploadFailedError => e
  @logger.error("S3 could not upload job script: \n#{e.message}")
  raise
end
end
```

创建运行场景的类。

```
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
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

```

创建一个 ETL 脚本，用于在作业运行期间 AWS Glue 提取、转换和加载数据。

```
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()

```

• 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的以下主题。

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

- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

操作

CreateCrawler

以下代码示例演示了如何使用 CreateCrawler。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateCrawler](#) 中的。

CreateJob

以下代码示例演示了如何使用 CreateJob。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateJob](#) 中的。

DeleteCrawler

以下代码示例演示了如何使用 DeleteCrawler。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DeleteCrawler](#) 中的。

DeleteDatabase

以下代码示例演示了如何使用 DeleteDatabase。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DeleteDatabase](#) 中的。

DeleteJob

以下代码示例演示了如何使用 DeleteJob。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteJob](#) 中的。

DeleteTable

以下代码示例演示了如何使用 DeleteTable。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteTable](#) 中的。

GetCrawler

以下代码示例演示了如何使用 GetCrawler。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetCrawler](#) 中的。

GetDatabase

以下代码示例演示了如何使用 GetDatabase。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [GetDatabase](#) 中的。

GetJobRun

以下代码示例演示了如何使用 GetJobRun。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetJobRun](#) 中的。

GetJobRuns

以下代码示例演示了如何使用 GetJobRuns。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [GetJobRuns](#) 中的。

GetTables

以下代码示例演示了如何使用 GetTables。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetTables](#) 中的。

ListJobs

以下代码示例演示了如何使用 ListJobs。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListJobs](#) 中的。

StartCrawler

以下代码示例演示了如何使用 StartCrawler。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [StartCrawler](#) 中的。

StartJobRun

以下代码示例演示了如何使用 StartJobRun。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [StartJobRun](#) 中的。

使用适用于 Ruby 的 SDK 的 IAM 示例

以下代码示例向您展示了如何使用 with IAM 来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK 基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [基本功能](#)
- [操作](#)

开始使用

开始使用 IAM

以下代码示例展示了如何开始使用 IAM。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListPolicies](#) 中的。

基本功能

了解基本功能

以下代码示例展示了如何创建用户并代入角色。

Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供者的联合身份验证，例如 [AWS IAM Identity Center](#)。

- 创建没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 S3 存储桶，然后清除资源。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
```

```
#
# @param duration [Integer] The number of seconds to wait.
def wait(duration)
  puts('Give AWS time to propagate resources...')
  sleep(duration)
end

# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_client.create_user(user_name: user_name).user
  @logger.info("Created demo user named #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info('Tried and failed to create demo user.')
  @logger.info("\t#{e.code}: #{e.message}")
  @logger.info("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
  @logger.info("Created accesskey pair for user #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create access keys for user #{user.user_name}.")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
```

```
def create_role(role_name, user)
  trust_policy = {
    Version: '2012-10-17',
    Statement: [{
      Effect: 'Allow',
      Principal: { 'AWS': user.arn },
      Action: 'sts:AssumeRole'
    }]
  }.to_json
  role = @iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: trust_policy
  ).role
  @logger.info("Created role #{role.role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create a role for the demo. Here's why: ")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  role
end

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy_document = {
    Version: '2012-10-17',
    Statement: [{
      Effect: 'Allow',
      Action: 's3:ListAllMyBuckets',
      Resource: 'arn:aws:s3::*'
    }]
  }.to_json
  policy = @iam_client.create_policy(
    policy_name: policy_name,
    policy_document: policy_document
  ).policy
  @iam_client.attach_role_policy(
    role_name: role.role_name,
    policy_arn: policy.arn
  )
end
```

```
)
  @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 })
  end
end
```

```

    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e

```

```
puts('Something went wrong with the demo.')
puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
$PROGRAM_NAME == __FILE__
```

• 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

操作

AttachRolePolicy

以下代码示例演示了如何使用 AttachRolePolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# 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
end
```

```
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
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考[AttachRolePolicy](#)中的。

AttachUserPolicy

以下代码示例演示了如何使用 AttachUserPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。


```
# Attaches a policy to a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_user(user_name, policy_arn)
  @iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error attaching policy to user: #{e.message}")
  false
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考[AttachUserPolicy](#)中的。

CreateAccessKey

以下代码示例演示了如何使用 CreateAccessKey。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

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

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
  end
end
```

```
@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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[CreateAccessKey](#)中的。

CreateAccountAlias

以下代码示例演示了如何使用 CreateAccountAlias。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
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
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[CreateAccountAlias](#)中的。

CreatePolicy

以下代码示例演示了如何使用 CreatePolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# 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
end
```

```
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
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [CreatePolicy](#) 中的。

CreateRole

以下代码示例演示了如何使用 CreateRole。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

```
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating role: #{e.message}")
  nil
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateRole](#) 中的。

CreateServiceLinkedRole

以下代码示例演示了如何使用 CreateServiceLinkedRole。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateServiceLinkedRole](#) 中的。

CreateUser

以下代码示例演示了如何使用 CreateUser。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateUser](#) 中的。

DeleteAccessKey

以下代码示例演示了如何使用 DeleteAccessKey。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

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

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
```

```
# @return [Boolean]
def create_access_key(user_name)
  response = @iam_client.create_access_key(user_name: user_name)
  access_key = response.access_key
  @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
  access_key
rescue Aws::IAM::Errors::LimitExceeded
  @logger.error('Error creating access key: limit exceeded. Cannot create more.')
  nil
rescue StandardError => e
  @logger.error("Error creating access key: #{e.message}")
  nil
end

# Deactivates an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def deactivate_access_key(user_name, access_key_id)
  @iam_client.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,
    status: 'Inactive'
  )
  true
rescue StandardError => e
  @logger.error("Error deactivating access key: #{e.message}")
  false
end

# Deletes an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def delete_access_key(user_name, access_key_id)
  @iam_client.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  true
rescue StandardError => e
```

```
@logger.error("Error deleting access key: #{e.message}")
  false
end
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DeleteAccessKey](#)中的。

DeleteAccountAlias

以下代码示例演示了如何使用 DeleteAccountAlias。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
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
  end
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteAccountAlias](#) 中的。

DeleteRole

以下代码示例演示了如何使用 DeleteRole。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteRole](#) 中的。

DeleteServerCertificate

以下代码示例演示了如何使用 DeleteServerCertificate。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DeleteServerCertificate](#)中的。

DeleteServiceLinkedRole

以下代码示例演示了如何使用 DeleteServiceLinkedRole。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DeleteServiceLinkedRole](#) 中的。

DeleteUser

以下代码示例演示了如何使用 DeleteUser。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteUser](#) 中的。

DeleteUserPolicy

以下代码示例演示了如何使用 DeleteUserPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteUserPolicy](#) 中的。

DetachRolePolicy

以下代码示例演示了如何使用 DetachRolePolicy。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# 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
  end
end
```

```
@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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DetachRolePolicy](#) 中的。

DetachUserPolicy

以下代码示例演示了如何使用 DetachUserPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [DetachUserPolicy](#) 中的。

GetAccountPasswordPolicy

以下代码示例演示了如何使用 `GetAccountPasswordPolicy`。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

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

```
    raise
  end
end
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考](#) [GetAccountPasswordPolicy](#) 中的。

GetPolicy

以下代码示例演示了如何使用 `GetPolicy`。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考](#) [GetPolicy](#) 中的。

GetRole

以下代码示例演示了如何使用 GetRole。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetRole](#) 中的。

GetUser

以下代码示例演示了如何使用 GetUser。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetUser](#) 中的。

ListAccessKeys

以下代码示例演示了如何使用 ListAccessKeys。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

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

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
    #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListAccessKeys](#)中的。

ListAccountAliases

以下代码示例演示了如何使用 ListAccountAliases。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListAccountAliases](#)中的。

ListAttachedRolePolicies

以下代码示例演示了如何使用 ListAttachedRolePolicies。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# 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
end
```

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListAttachedRolePolicies](#) 中的。

ListGroups

以下代码示例演示了如何使用 ListGroups。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [ListGroups](#) 中的。

ListPolicies

以下代码示例演示了如何使用 ListPolicies。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# 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
end
```

```
# @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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListPolicies](#) 中的。

ListRolePolicies

以下代码示例演示了如何使用 ListRolePolicies。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

```
[]  
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListRolePolicies](#)中的。

ListRoles

以下代码示例演示了如何使用 ListRoles。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListRoles](#) 中的。

ListSAMLProviders

以下代码示例演示了如何使用 ListSAMLProviders。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class SAMLProviderLister
  # Initializes the SAMLProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 SAMLProviders 中的 [列表](#)。

ListServerCertificates

以下代码示例演示了如何使用 ListServerCertificates。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
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
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListServerCertificates](#)中的。

ListUsers

以下代码示例演示了如何使用 ListUsers。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListUsers](#) 中的。

PutUserPolicy

以下代码示例演示了如何使用 PutUserPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。


```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [PutUserPolicy](#) 中的。

UpdateServerCertificate

以下代码示例演示了如何使用 UpdateServerCertificate。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[UpdateServerCertificate](#)中的。

UpdateUser

以下代码示例演示了如何使用 UpdateUser。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [UpdateUser](#) 中的。

使用适用于 Ruby 的 SDK 的 Kinesis 示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Kinesis 配合使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [无服务器示例](#)

无服务器示例

通过 Kinesis 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 Kinesis 流的记录而触发的事件。该函数检索 Kinesis 有效负载，将 Base64 解码，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 Kinesis 事件与 Lambda 结合使用。

```
# 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
```

```
end
```

通过 Kinesis 触发器报告 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 Kinesis 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告通过 Ruby 进行 Lambda Kinesis 批处理项目失败。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
      # immediately.
      # Lambda will immediately begin to retry processing from this failed item
      # onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
        ['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
```

```
{ batchItemFailures: batch_item_failures }  
end  
  
def get_record_data_async(payload)  
  data = Base64.decode64(payload['data']).force_encoding('utf-8')  
  # Placeholder for actual async work  
  sleep(1)  
  data  
end
```

AWS KMS 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用 `with` 来执行操作和实现常见场景 AWS KMS。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateKey

以下代码示例演示了如何使用 `CreateKey`。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateKey](#) 中的。

Decrypt

以下代码示例演示了如何使用 Decrypt。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Decrypt](#)。

Encrypt

以下代码示例演示了如何使用 Encrypt。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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*')
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Encrypt](#)。

ReEncrypt

以下代码示例演示了如何使用 ReEncrypt。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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*')
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ReEncrypt](#)中的。

使用适用于 Ruby 的 SDK 的 Lambda 示例

以下代码示例向您展示了如何使用 with Lambda 来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题


- [开始使用](#)
- [基本功能](#)
- [操作](#)
- [场景](#)
- [无服务器示例](#)

开始使用

开始使用 Lambda

以下代码示例展示了如何开始使用 Lambda。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [ListFunctions](#) 中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 IAM 角色和 Lambda 函数，然后上传处理程序代码。
- 使用单个参数来调用函数并获取结果。
- 更新函数代码并使用环境变量进行配置。
- 使用新参数来调用函数并获取结果。显示返回的执行日志。
- 列出账户函数，然后清除函数。

有关更多信息，请参阅[使用控制台创建 Lambda 函数](#)。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

为能够写入日志的 Lambda 函数设置必备的 IAM 权限。

```
# 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
```

定义一个 Lambda 处理程序，用以递增作为调用参数提供的数字。

```
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
```

将 Lambda 函数压缩到部署程序包中。

```
# 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
```

新建 Lambda 函数。

```
# 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'
      }
    }
  })
end
```

```
        })
    @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
```

使用可选的运行时参数调用 Lambda 函数。

```
# 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
```

更新 Lambda 函数的配置，以注入新的环境变量。

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

```

    })
    @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::Writers::Errors::WaiterFailed => e
      @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
    end
  end

```

使用包含不同代码的不同部署程序包更新 Lambda 函数的代码。

```

# 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::Writers::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
end

```

使用内置分页工具列出所有现有的 Lambda 函数。

```
# 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
```

删除特定的 Lambda 函数。

```
# 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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的以下主题。
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

操作

CreateFunction

以下代码示例演示了如何使用 CreateFunction。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  # code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
      role: role_arn.to_s,
      function_name: function_name,
      handler: handler_name,
      runtime: 'ruby2.7',
      code: {
        zip_file: deployment_package
      },
    },
```

```
environment: {
  variables: {
    'LOG_LEVEL' => 'info'
  }
}
}))
@lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
  w.max_attempts = 5
  w.delay = 5
end
response
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error creating #{function_name}:\n #{e.message}")
rescue Aws::Waiters::Errors::WaiterFailed => e
  @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateFunction](#) 中的。

DeleteFunction

以下代码示例演示了如何使用 DeleteFunction。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DeleteFunction](#)中的。

GetFunction

以下代码示例演示了如何使用 GetFunction。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[GetFunction](#)中的。

Invoke

以下代码示例演示了如何使用 Invoke。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Invoke](#)。

ListFunctions

以下代码示例演示了如何使用 ListFunctions。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
  end
end
```

```
end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListFunctions](#)中的。

UpdateFunctionCode

以下代码示例演示了如何使用 UpdateFunctionCode。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [UpdateFunctionCode](#) 中的。

UpdateFunctionConfiguration

以下代码示例演示了如何使用 UpdateFunctionConfiguration。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考 UpdateFunctionConfiguration](#) 中的。

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。

- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

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

无服务器示例

使用 Lambda 函数连接到 Amazon RDS 数据库

以下代码示例显示如何实现连接到 RDS 数据库的 Lambda 函数。该函数发出一个简单的数据库请求并返回结果。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

在 Lambda 函数中使用 Ruby 连接到 Amazon RDS 数据库。

```
# 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
```

通过 Kinesis 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 Kinesis 流的记录而触发的事件。该函数检索 Kinesis 有效负载，将 Base64 解码，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 Kinesis 事件与 Lambda 结合使用。

```
# 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
```

```
end
```

通过 DynamoDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DynamoDB 流的记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 DynamoDB 事件与 Lambda 结合使用。

```
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
```

通过 Amazon DocumentDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DocumentDB 更改流的记录而触发的事件。该函数检索 DocumentDB 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Ruby 将 Amazon DocumentDB 事件与 Lambda 结合使用。

```
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
```

通过 Amazon MSK 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 Amazon MSK 集群的记录而触发的事件。该函数检索 MSK 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 Amazon MSK 事件与 Lambda 结合使用。

```
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
```

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 S3 事件与 Lambda 结合使用。

```
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
```

通过 Amazon SNS 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 SNS 主题的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 SNS 事件与 Lambda 结合使用。

```
# 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
```

通过 Amazon SQS 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 SNS 队列的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Ruby 将 SQS 事件与 Lambda 结合使用。

```
# 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
```

通过 Kinesis 触发器报告 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 Kinesis 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告通过 Ruby 进行 Lambda Kinesis 批处理项目失败。

```
# 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
```

通过 DynamoDB 触发器报告 Lambda 函数批处理项目失败

以下代码示例演示如何为接收来自 DynamoDB 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Ruby 通过 Lambda 进行 DynamoDB 批处理项目失败。

```
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
```

报告使用 Amazon SQS 触发器进行 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 SQS 队列的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Ruby 进行 Lambda SQS 批处理项目失败。

```
# 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
```

使用适用于 Ruby 的 SDK 的 Amazon MSK 示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Amazon MSK 配合使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [无服务器示例](#)

无服务器示例

通过 Amazon MSK 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 Amazon MSK 集群的记录而触发的事件。该函数检索 MSK 有效负载，并记录下记录内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 Amazon MSK 事件与 Lambda 结合使用。

```
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
```

使用适用于 Ruby 的 SDK 的 Amazon Polly 示例

以下代码示例向您展示了如何在 Amazon Polly 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

DescribeVoices

以下代码示例演示了如何使用 DescribeVoices。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DescribeVoices](#) 中的。

ListLexicons

以下代码示例演示了如何使用 ListLexicons。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListLexicons](#) 中的。

SynthesizeSpeech

以下代码示例演示了如何使用 SynthesizeSpeech。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [SynthesizeSpeech](#) 中的。

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

使用适用于 Ruby 的 SDK 的 Amazon RDS 示例

以下代码示例向您展示了如何在 Amazon RDS 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [操作](#)
- [无服务器示例](#)

开始使用

开始使用 Amazon RDS

以下代码示例展示了如何开始使用 Amazon RDS。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-rds'
require 'logger'

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

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

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
          (#{instance.db_instance_status})")
      end
    end
  end
end

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 DBInstances 中的 [描述](#)。

操作

CreateDBSnapshot

以下代码示例演示了如何使用 CreateDBSnapshot。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 DBSnapshot 在 适用于 Ruby 的 AWS SDK API 参考中 [创建](#)。

DescribeDBInstances

以下代码示例演示了如何使用 DescribeDBInstances。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 DBInstances 中的 [描述](#)。

DescribeDBParameterGroups

以下代码示例演示了如何使用 DescribeDBParameterGroups。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考中的[描述DBParameter群组](#)。

DescribeDBParameters

以下代码示例演示了如何使用 DescribeDBParameters。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考DBParameters中的[描述](#)。

DescribeDBSnapshots

以下代码示例演示了如何使用 DescribeDBSnapshots。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end
```

```
instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考DBSnapshots中的[描述](#)。

无服务器示例

使用 Lambda 函数连接到 Amazon RDS 数据库

以下代码示例显示如何实现连接到 RDS 数据库的 Lambda 函数。该函数发出一个简单的数据库请求并返回结果。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

在 Lambda 函数中使用 Ruby 连接到 Amazon RDS 数据库。

```
# 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_clear_text_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
```

使用适用于 Ruby 的 SDK 的 Amazon S3 示例

以下代码示例向您展示了如何在 Amazon S3 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [开始使用](#)
- [基本功能](#)
- [操作](#)
- [场景](#)
- [无服务器示例](#)

开始使用

开始使用 Amazon S3

以下代码示例显示了如何开始使用 Amazon S3。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListBuckets](#) 中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建桶并将文件上载到其中。
- 从桶中下载对象。
- 将对象复制到存储桶中的子文件夹。

- 列出存储桶中的对象。
- 删除存储桶及其对象。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

```
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
```

```
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
  answer = gets.chomp.downcase
```

```
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
  rescue Aws::Errors::ServiceError
    puts('Something went wrong with the demo!')
  end

  run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
  __FILE__
end
```

• 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的以下主题。

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

操作

CopyObject

以下代码示例演示了如何使用 CopyObject。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

复制对象。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
```

```

    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
    #{e.message}"
  end
end
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__

```

复制对象并向目标对象添加服务器端加密。

```

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
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CopyObject](#) 中的。

CreateBucket

以下代码示例演示了如何使用 CreateBucket。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__

```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateBucket](#) 中的。

DeleteBucket

以下代码示例演示了如何使用 DeleteBucket。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteBucket](#) 中的。

DeleteBucketCors

以下代码示例演示了如何使用 DeleteBucketCors。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
  # existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
```

```
# @return [Boolean] True if the CORS rules were deleted; otherwise, false.
def delete_cors
  @bucket_cors.delete
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end
end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[DeleteBucketCors](#)中的。

DeleteBucketPolicy

以下代码示例演示了如何使用 DeleteBucketPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end

```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteBucketPolicy](#) 中的。

DeleteObjects

以下代码示例演示了如何使用 DeleteObjects。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

# 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

```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteObjects](#) 中的。

GetBucketCors

以下代码示例演示了如何使用 GetBucketCors。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetBucketCors](#) 中的。

GetBucketPolicy

以下代码示例演示了如何使用 GetBucketPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [GetBucketPolicy](#) 中的。

GetObject

以下代码示例演示了如何使用 GetObject。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

获取对象。

```
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
end
```

```
puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to  
#{target_path}."  
end  
  
run_demo if $PROGRAM_NAME == __FILE__
```

获取对象并报告其服务器端加密状态。

```
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  
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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[GetObject](#)中的。

HeadObject

以下代码示例演示了如何使用 HeadObject。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
    Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[HeadObject](#)中的。

ListBuckets

以下代码示例演示了如何使用 ListBuckets。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListBuckets](#) 中的。

ListObjectsV2

以下代码示例演示了如何使用 ListObjectsV2。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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

# 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__
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考中的 [ListObjectsV2](#)。

PutBucketCors

以下代码示例演示了如何使用 PutBucketCors。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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}"
  end
end
```

```
    false
  end

end
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[PutBucketCors](#)中的。

PutBucketPolicy

以下代码示例演示了如何使用 PutBucketPolicy。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [PutBucketPolicy](#) 中的。

PutBucketWebsite

以下代码示例演示了如何使用 PutBucketWebsite。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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 }
      }
    )
  end
end
```

```
    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
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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [PutBucketWebsite](#) 中的。

PutObject

以下代码示例演示了如何使用 PutObject。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

使用托管上传工具 (Object.upload_file) 上传文件。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
```

```
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 `Object.put` 上传文件。

```
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__

```

使用 `Object.put` 上传文件并添加服务器端加密。

```

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

# 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__
```


- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [PutObject](#) 中的。

场景

创建预签名 URL

以下代码示例演示了如何为 Amazon S3 创建预签名 URL 以及如何上传对象。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end

run_demo if $PROGRAM_NAME == __FILE__
```

无服务器示例

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 S3 事件与 Lambda 结合使用。

```
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
```

使用适用于 Ruby 的 SDK 的 Amazon SES 示例

以下代码示例向您展示了如何在 Amazon SES 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetIdentityVerificationAttributes

以下代码示例演示了如何使用 GetIdentityVerificationAttributes。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 [适用于 Ruby 的 AWS SDK API 参考](#) [GetIdentityVerificationAttributes](#) 中的。

ListIdentities

以下代码示例演示了如何使用 ListIdentities。

适用于 Ruby 的 SDK

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListIdentities](#) 中的。

SendEmail

以下代码示例演示了如何使用 SendEmail。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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>.'
```

```
    text: {
      charset: encoding,
      data: textbody
    }
  },
  subject: {
    charset: encoding,
    data: subject
  }
},
source: sender
# Uncomment the following line to use a configuration set.
# configuration_set_name: configsetname,
)

puts "Email sent to #{recipient}"

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

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[SendEmail](#)中的。

VerifyEmailIdentity

以下代码示例演示了如何使用 VerifyEmailIdentity。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'
```

```
# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts "Email sent to #{recipient}"

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

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考[VerifyEmailIdentity](#)中的。

使用适用于 Ruby 的 SDK 的 Amazon SES API v2 示例

以下代码示例向您展示了如何在 Amazon SES API v2 中 适用于 Ruby 的 AWS SDK 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题


- [操作](#)

操作

SendEmail

以下代码示例演示了如何使用 SendEmail。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [SendEmail](#) 中的。

使用适用于 Ruby 的 SDK 的 Amazon SNS 示例

以下代码示例向您展示了如何在 Amazon SNS 中使用来执行操作和实现常见场景。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [无服务器示例](#)

操作

CreateTopic

以下代码示例演示了如何使用 CreateTopic。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
end
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-topic.html>。
- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[CreateTopic](#)中的。

ListSubscriptions

以下代码示例演示了如何使用 ListSubscriptions。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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}"
  end
end
```

```
    exit 1
  end
end
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-show-subscriptions.html>。
- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ListSubscriptions](#)中的。

ListTopics

以下代码示例演示了如何使用 ListTopics。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)
end
```

```
puts 'The bucket was not created. Stopping program.'
exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-show-topics.html>。
- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考 [ListTopics](#) 中的。

Publish

以下代码示例演示了如何使用 Publish。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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

# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-send-message.html>。
- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考中的 [Publish](#)。

SetTopicAttributes

以下代码示例演示了如何使用 SetTopicAttributes。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-enable-resource.html>。
- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[SetTopicAttributes](#)中的。

Subscribe

以下代码示例演示了如何使用 Subscribe。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

将电子邮件地址订阅到主题。

```
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
```

- 有关更多信息，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-subscription.html>。
- 有关 API 详细信息，请参阅《适用于 Ruby 的 AWS SDK API Reference》中的 [Subscribe](#)。

无服务器示例

通过 Amazon SNS 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 SNS 主题的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 SNS 事件与 Lambda 结合使用。

```
# 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
```

```
end
```

使用适用于 Ruby 的 SDK 的 Amazon SQS 示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Amazon SQS 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [无服务器示例](#)

操作

ChangeMessageVisibility

以下代码示例演示了如何使用 ChangeMessageVisibility。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ChangeMessageVisibility](#) 中的。

CreateQueue

以下代码示例演示了如何使用 CreateQueue。

适用于 Ruby 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [CreateQueue](#) 中的。

DeleteQueue

以下代码示例演示了如何使用 DeleteQueue。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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)
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DeleteQueue](#) 中的。

ListQueues

以下代码示例演示了如何使用 ListQueues。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListQueues](#) 中的。

ReceiveMessage

以下代码示例演示了如何使用 ReceiveMessage。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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__

```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[ReceiveMessage](#)中的。

SendMessage

以下代码示例演示了如何使用 SendMessage。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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__
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [SendMessage](#) 中的。

SendMessageBatch

以下代码示例演示了如何使用 SendMessageBatch。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [SendMessageBatch](#) 中的。

无服务器示例

通过 Amazon SQS 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 SNS 队列的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Ruby 将 SQS 事件与 Lambda 结合使用。

```
# 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
```

报告使用 Amazon SQS 触发器进行 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 SQS 队列的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Ruby 进行 Lambda SQS 批处理项目失败。

```
# 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 使用适用于 Ruby 的 SDK 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS STS。适用于 Ruby 的 AWS SDK

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

AssumeRole

以下代码示例演示了如何使用 AssumeRole。

适用于 Ruby 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
# 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
```

- 有关 API 的详细信息，请参阅适用于 Ruby 的 AWS SDK API 参考[AssumeRole](#)中的。

使用适用于 Ruby 的 SDK 的 Amazon Textract 示例

以下代码示例向您展示了如何在 Amazon Textract 中 适用于 Ruby 的 AWS SDK 使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [场景](#)

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

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

使用适用于 Ruby 的 SDK 的 Amazon Translate 示例

以下代码示例向您展示了如何使用 适用于 Ruby 的 AWS SDK 与 Amazon Translate 配合使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [场景](#)

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

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

从 Ruby 版 AWS SDK 版本 1 或 2 迁移到 AWS 适用于 Ruby 的 SDK 版本 3

本主题包含的详细信息可帮助您从适用于 Ruby 的 AWS SDK 的版本 1 或 2 迁移到版本 3。

Side-by-side 用法

没有必要将适用于 Ruby 的 AWS SDK 的版本 1 或 2 替换为版本 3。您可以在同一应用程序中一起使用它们。有关更多信息，请参阅[此博客文章](#)。

下面是一个快速示例。

```
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
```

您无需重写现有的工作版本 1 或 2 代码即可开始使用版本 3 开发工具包。有效的迁移策略是仅针对版本 3 开发工具包编写新代码。

一般区别

版本 3 在一个重要方面与版本 2 不同。

- 每项服务都作为单独的 Gem 提供。

版本 2 在多个重要方面与版本 1 不同。

- 不同的根命名空间：Aws 与 AWS。这样可以 side-by-side 使用。
- Aws.config – 现在是 vanilla Ruby 哈希，而不是方法。
- 严格的构造函数选项 - 在版本 1 开发工具包中构造客户端或资源对象时，会忽略未知构造函数选项。在版本 2 中，未知构造函数选项会触发 ArgumentError。例如：

```
# 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
```

客户端区别

版本 2 和版本 3 中的客户端类之间没有区别。

在版本 1 和版本 2 之间，客户端类具有最少的外部区别。在构造客户端后，许多服务客户端都将具有兼容的接口。一些重要区别：

- `Aws::S3::Client`：版本 1 Amazon S3 客户端类是手动编码的。版本 2 是从服务模型生成的。方法名称和输入在版本 2 中大不相同。
- `Aws::EC2::Client` - 版本 2 对输出列表使用复数名称，版本 1 使用后缀 `_set`。例如：

```
# 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` – 版本 2 使用结构化响应，而版本 1 使用 vanilla Ruby 哈希。
- 服务类重命名 – 版本 2 对多个服务使用不同的名称：
 - `AWS::SimpleWorkflow` 变成 `Aws::SWF`
 - `AWS::ELB` 变成 `Aws::ElasticLoadBalancing`
 - `AWS::SimpleEmailService` 变成 `Aws::SES`
- 客户端配置选项：一些版本 1 配置选项在版本 2 中被重命名。其他选项被删除或被替换。下面是主要的更改：
 - `:use_ssl` 已删除。版本 2 到处使用 SSL。要禁用 SSL，您必须配置使用 `:endpoint` 的 `http://`。
 - `:ssl_ca_file` 现在是 `:ssl_ca_bundle`

- `:ssl_ca_path` 现在是 `:ssl_ca_directory`
- 增加了 `:ssl_ca_store`。
- `:endpoint` 现在必须是完全限定的 HTTP 或 HTTPS URI 而非主机名。
- 为每个服务删除了 `:*_port` 选项，现在替换为 `:endpoint`。
- `:user_agent_prefix` 现在是 `:user_agent_suffix`

资源区别

版本 2 和版本 3 中的资源接口之间没有区别。

版本 1 和版本 2 中的资源接口之间存在显著区别。版本 1 是完全手动编码的，而版本 2 资源接口是从模型生成的。版本 2 资源接口明显更一致。一些系统区别包括：

- 单独的资源类：在版本 2 中，服务名称是模块而不是类。在此模块中，它为资源接口：

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- 参考资源 – 版本 2 开发工具包将集合和单个资源 getter 分成两个不同的方法：

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- 批量操作：在版本 1 中，所有批量操作都是手动编码的实用程序。在版本 2 中，许多批量操作是通过 API 自动生成的批处理操作。版本 2 批处理接口与版本 1 大不相同。

AWS 适用于 Ruby 的 SDK 的安全性

云安全性一直是 Amazon Web Services (AWS) 的重中之重。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。安全是双方共同承担 AWS 的责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性。

云安全 — AWS 负责保护运行 AWS 云中提供的所有服务的基础架构，并为您提供可以安全使用的服务。我们的安全责任是重中之重 AWS，作为[AWS 合规计划](#)的一部分，第三方审计师定期测试和验证我们安全的有效性。

云端安全 — 您的责任由 AWS 服务 您使用的内容以及其他因素决定，包括数据的敏感性、组织的要求以及适用的法律和法规。

主题

- [适用于 Ruby 的 AWS SDK 中的数据保护](#)
- [适用于 Ruby 的 AWS SDK 的身份和访问管理](#)
- [适用于 Ruby 的 AWS SDK 的合规性验证](#)
- [AWS 适用于 Ruby 的 SDK 的弹性](#)
- [适用于 Ruby 的 AWS SDK 的基础架构安全](#)
- [在 Ruby 的 S AWS DK 中强制使用最低 TLS 版本](#)
- [亚马逊 S3 加密客户端迁移 \(从 V1 到 V2 \)](#)
- [亚马逊 S3 加密客户端迁移 \(从 V2 到 V3 \)](#)

适用于 Ruby 的 AWS SDK 中的数据保护

分 AWS [担责任模型](#)适用于中的数据保护。如本模型所述 AWS，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 [AWS Shared Responsibility Model and GDPR](#) 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。

- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅 AWS CloudTrail 用户指南中的 [使用跟 CloudTrail 踪](#)。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》<https://aws.amazon.com/compliance/fips/>。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或以其他 AWS 服务方式使用控制台 AWS CLI、API 或 AWS SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

适用于 Ruby 的 AWS SDK 的身份和访问管理

AWS Identity and Access Management (IAM) 是一项 Amazon Web Services (AWS) 服务，可帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 AWS 服务资源。您可以使用 IAM AWS 服务，无需支付额外费用。

要使用 AWS SDK for Ruby 进行访问 AWS，你需要一个 AWS 账户和 AWS 证书。为了提高 AWS 账户的安全性，我们建议您使用 IAM 用户 来提供访问凭证，而不使用 AWS 账户凭证。

有关使用 IAM 的详细信息，请参阅 [IAM](#)。

有关 IAM 用户以及它们对于账户的安全性为何十分重要的概览，请参阅 [Amazon Web Services 一般参考](#) 中的 [AWS 安全凭证](#)。

AWS 适用于 Ruby 的 SDK 通过其支持的特定亚马逊 Web Services (AWS) 服务遵循 [分担责任模式](#)。有关 AWS 服务 安全信息，请参阅 [AWS 服务 安全文档页面](#) [AWS 服务，这些信息属于 AWS 合规计划的合规工作范围](#)。

适用于 Ruby 的 AWS SDK 的合规性验证

AWS 适用于 Ruby 的 SDK 通过其支持的特定亚马逊 Web Services (AWS) 服务遵循 [分担责任模式](#)。有关 AWS 服务 安全信息，请参阅 [AWS 服务 安全文档页面](#) [AWS 服务，这些信息属于 AWS 合规计划的合规工作范围](#)。

Amazon Web Services (AWS) 服务的安全性与合规性由第三方审计机构作为多项 AWS 合规计划的一部分进行评估。其中包括 SOC、PCI、FedRAMP、HIPAA 等。AWS 在“[按合规计划划分的范围 AWS 服务 内的 AWS 服务](#)”中提供了经常更新的特定合规计划列表。

您可以使用第三方审计报告进行下载 AWS Artifact。有关更多信息，请参阅在 [Artifact 中 AWS 下载报告](#)。

有关 AWS 合规计划的更多信息，请参阅[AWS 合规计划](#)。

使用 AWS SDK for Ruby 访问时，您的合规责任取决于您的数据的敏感度、组织的合规目标以及适用的法律和法规。AWS 服务 如果您在使用时必须符合 HIPAA、PCI 或 FedRAMP 等标准，请提供资源来帮助：AWS 服务 AWS

- [安全与合规性快速入门指南](#) — 部署指南，讨论架构注意事项，并提供在上部署以安全为重点和以合规为重点的基准环境的步骤。AWS
- [符合 HIPAA 要求的服务参考](#)：列出符合 HIPAA 要求的服务。并非所有 AWS 服务 人都符合 HIPAA 资格。
- [AWS 合规资源](#) — 可能适用于您所在行业和所在地区的工作手册和指南的集合。
- [AWS Config](#) — 一项评估您的资源配置在多大程度上符合内部实践、行业准则和法规的服务。
- [AWS Security Hub](#) — 全面了解您的安全状态 AWS ，可帮助您检查自己是否符合安全行业标准和最佳实践。

AWS 适用于 Ruby 的 SDK 的弹性

Amazon Web Services (AWS) 全球基础设施是围绕 AWS 区域 可用区构建的。

AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。

利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础架构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅[AWS 全球基础设施](#)。

AWS 适用于 Ruby 的 SDK 通过其支持的特定亚马逊 Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务 安全信息，请参阅[AWS 服务 安全文档页面AWS 服务，这些信息属于 AWS 合规计划的合规工作范围](#)。

适用于 Ruby 的 AWS SDK 的基础架构安全

AWS 适用于 Ruby 的 SDK 通过其支持的特定亚马逊 Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)[AWS 服务，这些信息属于 AWS 合规计划的合规工作范围](#)。

有关 AWS 安全流程的信息，请参阅 [AWS：安全流程概述](#) 白皮书。

在 Ruby 的 S AWS DK 中强制使用最低 TLS 版本

AWS 适用于 Ruby AWS 的 SDK 与之间的通信使用安全套接字层 (SSL) 或传输层安全 (TLS) 进行保护。所有版本的 SSL 以及 1.2 之前的 TLS 版本都存在漏洞，这些漏洞可能会危及您与之通信的安全性 AWS。因此，您应确保将适用于 Ruby 的 AWS SDK 与支持 TLS 1.2 或更高版本的 Ruby 版本一起使用。

Ruby 使用 OpenSSL 库来保护 HTTP 连接。通过系统[程序包管理器](#) (yum、apt 等)、[官方安装程序](#)或 Ruby [管理器](#) (rbenv、RVM 等) 安装的受支持 Ruby 版本 (1.9.3 及更高版本) 通常使用 OpenSSL 1.0.1 或更高版本，这些版本支持 TLS 1.2。

当与 OpenSSL 1.0.1 或更高版本的 Ruby 支持版本一起使用时，AWS 适用于 Ruby 的 SDK 更喜欢 TLS 1.2，并使用客户端和服务端都支持的最新版本的 SSL 或 TLS。这始终至少是 TLS 1.2 AWS 服务。(此 SDK 将使用 `use_ssl=true` 的 `Ruby Net::HTTP` 类。)

检查 OpenSSL 版本

要确保您安装的 Ruby 使用的是 OpenSSL 1.0.1 或更高版本，请输入以下命令。

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

获取 OpenSSL 版本的另一种方法是直接查询 `openssl` 可执行文件。首先，使用以下命令查找适当的可执行文件。

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

输出应包含指示 OpenSSL 安装位置的 `--with-openssl-dir=/path/to/openssl`。记下此路径。要检查 OpenSSL 的版本，请输入以下命令。

```
cd /path/to/openssl
```

```
bin/openssl version
```

后一种方法可能不适用于所有 Ruby 安装。

升级 TLS 支持

如果您的 Ruby 安装使用的是 1.0.1 版之前的 OpenSSL 版本，请使用系统程序包管理器、Ruby 安装程序或 Ruby 管理器升级 Ruby 或 OpenSSL 安装，如 Ruby 的[安装指南](#)中所述。如果您要[从源代码](#)安装 Ruby，请先安装[最新的 OpenSSL](#)，然后在运行 `./configure` 时通过 `--with-openssl-dir=/path/to/upgraded/openssl`。

亚马逊 S3 加密客户端迁移 (从 V1 到 V2)

Note

如果您使用的是 S3 加密客户端的 V2，并且想要迁移到 V3，请参阅[亚马逊 S3 加密客户端迁移 \(从 V2 到 V3 \)](#)。

此主题介绍了如何将应用程序从 Amazon Simple Storage Service (Amazon S3) 加密客户端的版本 1 (V1) 迁移到版本 2 (V2)，并确保应用程序在整个迁移过程中的可用性。

迁移概述

此迁移分为两个阶段：

1. 更新现有客户端以读取新格式。首先，将适用于 Ruby 的 AWS SDK 的更新版本部署到您的应用程序。这将允许现有 V1 加密客户端解密由新的 V2 客户端写入的对象。如果您的应用程序使用多个 SDK AWS SDKs，则必须单独升级每个 SDK。
2. 将加密和解密客户端迁移到 V2。一旦所有 V1 加密客户端都能读取新格式，就可以将现有加密和解密客户端迁移到各自的 V2 版本。

更新现有客户端以读取新格式

V2 加密客户端使用旧版本客户端不支持的加密算法。迁移的第一步是将 V1 解密客户端更新到最新 SDK 版本。完成此步骤后，应用程序的 V1 客户端就能够解密由 V2 加密客户端加密的对象。有关适用于 Ruby 的 AWS SDK 的每个主要版本，请参见下文。

更新 AWS 适用于 Ruby 的 SDK 版本 3

版本 3 是适用于 Ruby 的 AWS 软件开发工具包的最新版本。要完成此迁移，您需要使用版本 1.76.0 或更高版本的 `aws-sdk-s3` Gem。

从命令行安装

对于安装 `aws-sdk-s3` Gem 的项目，请使用版本选项来验证是否安装了不低于 1.76.0 的版本。

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

使用 Gemfile

对于使用 Gemfile 管理依赖项的项目，请将 `aws-sdk-s3` Gem 的最低版本设置为 1.76.0。例如：

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. 修改 Gemfile。
2. 运行 `bundle update aws-sdk-s3`。要验证您的版本，请运行 `bundle info aws-sdk-s3`。

升级 AWS 适用于 Ruby 的 SDK 版本 2

适用于 Ruby 的 AWS SDK 第 2 版将于 2021 年 11 月 21 日进入[维护模式](#)。要完成此迁移，您需要使用版本 2.11.562 或更高版本的 `aws-sdk` Gem。

从命令行安装

对于安装 `aws-sdk` Gem 的项目，请从命令行使用版本选项来验证是否安装了不低于 2.11.562 的版本。

```
gem install aws-sdk -v '>= 2.11.562'
```

使用 Gemfile

对于使用 Gemfile 管理依赖项的项目，请将 `aws-sdk` Gem 的最低版本设置为 2.11.562。例如：

```
gem 'aws-sdk', '>= 2.11.562'
```

1. 修改 Gemfile。如果存在 Gemfile.lock 文件，请将其删除或更新。
2. 运行 `bundle update aws-sdk`。要验证您的版本，请运行 `bundle info aws-sdk`。

将加密和解密客户端迁移到 V2

更新客户端以读取新的加密格式后，您可以将应用程序更新到 V2 加密和解密客户端。以下步骤展示了如何成功地将代码从 V1 迁移到 V2。

在更新代码以使用 V2 加密客户端之前，确保您已按照上述步骤操作且使用的是 `aws-sdk-s3` Gem 版本 2.11.562 或更高版本。

Note

使用 AES-GCM 解密时，在开始使用解密的数据之前，请通读整个对象。这是为了验证自加密以来是否未对对象进行过修改。

配置 V2 加密客户端

`EncryptionV2::Client` 需要额外的配置。有关详细的配置信息，请参阅 [EncryptionV2::Client 文档](#) 或本主题后面提供的示例。

1. 必须在构造客户端时指定密钥包装方法和内容加密算法。在创建新的 `EncryptionV2::Client` 时，需要为 `key_wrap_schema` 和 `content_encryption_schema` 提供值。

`key_wrap_schema`-如果您正在使用 AWS KMS，则必须将其设置为 `:kms_context`。如果您使用的是对称 (AES) 密钥，则必须将其设置为 `:aes_gcm`。如果您使用的是非对称 (RSA) 密钥，则必须将其设置为 `:rsa_oaep_sha1`。

`content_encryption_schema` : 必须将其设置为 `:aes_gcm_no_padding`。

2. 必须在构造客户端时指定 `security_profile`。在创建新的 `EncryptionV2::Client` 时，需要为 `security_profile` 提供值。`security_profile` 参数决定了是否支持读取使用旧版 V1 `Encryption::Client` 写入的对象。有两个值 `:v2` 和 `:v2_and_legacy`。要支持迁移，请将 `security_profile` 设置为 `:v2_and_legacy`。`:v2` 仅适用于新应用程序开发。

3. AWS KMS key 默认情况下强制使用 ID。在 V1 中 `Encryption::Client`，`kms_key_id` 用于创建客户端的未提供给。AWS KMS Decrypt call AWS KMS 可以从元数据中获取

这些信息并将其添加到对称密文 blob 中。在 V2 中，`EncryptionV2::Client`，`kms_key_id` 被传递给 AWS KMS Decrypt 调用，如果它与用于加密对象的密钥不匹配，则调用将失败。如果代码以前依赖于不设置特定的 `kms_key_id`，则要么在创建客户端时设置 `kms_key_id: :kms_allow_decrypt_with_any_cmek`，要么在调用 `get_object` 时设置 `kms_allow_decrypt_with_any_cmek: true`。

示例：使用对称 (AES) 密钥

迁移前

```
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)
```

迁移后

```
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
```

示例：与 `kms_key_id` 一起使用

迁移前

```
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)
```

迁移后

```
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
```

示例：在没有 kms_key_id 的情况下使用 AWS KMS

迁移前

```
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)
```

迁移后

```
client = Aws::S3::EncryptionV2::Client.new(  
  kms_key_id: kms_key_id,  
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys  
  content_encryption_schema: :aes_gcm_no_padding,  
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by  
  the V1 encryption client  
)  
client.put_object(bucket: bucket, key: key, body: secret_data) # No changes  
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:  
  true) # To allow decrypting with any cmk
```

迁移后替代方案

如果仅使用 S3 加密客户端读取和解密（从不写入和加密）对象，请使用此代码。

```
client = Aws::S3::EncryptionV2::Client.new(  
  kms_key_id: :kms_allow_decrypt_with_any_cmk, # set kms_key_id to allow all get_object  
  requests to use any cmk  
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys  
  content_encryption_schema: :aes_gcm_no_padding,  
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by  
  the V1 encryption client  
)  
resp = client.get_object(bucket: bucket, key: key) # No change
```

亚马逊 S3 加密客户端迁移 (从 V2 到 V3)

Note

如果您使用的是 S3 加密客户端的 V1，则必须先迁移到 V2，然后再迁移到 V3。 [亚马逊 S3 加密客户端迁移 \(从 V1 到 V2 \)](#) 有关从 V1 迁移到 V2 的说明，请参阅。

本主题介绍如何将您的应用程序从亚马逊简单存储服务 (Amazon S3) Simple Service 加密客户端的版本 2 (V2) 迁移到版本 3 (V3)，并确保应用程序在整个迁移过程中的可用性。V3 引入了 AES GCM 以及密钥承诺和承诺政策，以增强安全性并防止数据密钥被篡改。

迁移概述

Amazon S3 加密客户端第 3 版引入了带有密钥承诺的 AES GCM，以增强安全性。这种新的加密算法可防止数据密钥被篡改，并确保加密数据的完整性。迁移到 V3 需要仔细规划，以便在整个过程中保持应用程序可用性和数据可访问性。

此迁移分为两个阶段：

1. 更新现有客户端以读取新格式。首先，将适用于 Ruby 的 AWS SDK 的更新版本部署到您的应用程序。这将允许现有的 V2 加密客户端解密新 V3 客户端写入的对象。如果您的应用程序使用多个 SDK AWS SDKs，则必须单独升级每个 SDK。
2. 将加密和解密客户端迁移到 V3。一旦所有 V2 加密客户端都能读取新格式，就可以将现有的加密和解密客户端迁移到各自的 V3 版本。这包括配置承诺策略和更新代码以使用新的客户端配置选项。

如果您尚未从 V1 迁移到 V2，则必须先完成迁移。 [亚马逊 S3 加密客户端迁移 \(从 V1 到 V2 \)](#) 有关从 V1 迁移到 V2 的详细说明，请参阅。

了解 V3 的功能

Amazon S3 加密客户端第 3 版引入了两个关键安全功能：承诺策略和带密钥承诺的 AES GCM。了解这些功能对于规划迁移策略和确保加密数据的安全性至关重要。

承诺政策

承诺策略控制加密客户端在加密和解密操作期间如何处理密钥承诺。密钥承诺可确保加密数据只能使用用于加密数据的精确密钥进行解密，从而防止某些类型的加密攻击。

V3 加密客户端支持三个承诺策略选项：

FORBID_ENCRYPT_ALLOW_DECRYPT

此策略对没有密钥承诺的对象进行加密，并允许使用和不使用密钥承诺对两个对象进行解密。

- 加密行为：使用与 V2 相同的算法套件对对象进行加密，无需密钥承诺。
- 解密行为：可以解密使用或不使用密钥承诺加密的对象。
- 安全影响：此政策不强制执行密钥承诺，可能允许篡改。使用此策略加密的对象不会受益于密钥承诺的增强安全保护。只有在需要保持与 V2 加密行为的兼容性时，才在迁移期间使用此策略。
- 版本兼容性：S3 加密客户端的所有 V2 和 V3 实现均可读取使用此策略加密的对象。

REQUIRE_ENCRYPT_ALLOW_DECRYPT

此策略使用密钥承诺对对象进行加密，并允许使用和不使用密钥承诺对两个对象进行解密。

- 加密行为：使用带有密钥承诺的 AES GCM 使用密钥承诺对对象进行加密。
- 解密行为：可以解密使用或不使用密钥承诺加密的对象，从而提供向后兼容性。
- 安全影响：新对象受益于密钥承诺保护，而没有密钥承诺的现有对象仍然可以读取。这样可以在迁移期间在安全性和向后兼容性之间取得平衡。
- 版本兼容性：使用此策略加密的对象只能由 V3 和 S3 加密客户端的最新 V2 实现进行读取。

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

此策略对使用密钥承诺的对象进行加密，并且仅允许解密使用密钥承诺加密的对象。

- 加密行为：使用带有密钥承诺的 AES GCM 使用密钥承诺对对象进行加密。
- 解密行为：只能解密使用密钥承诺加密的对象。尝试在没有密钥承诺的情况下解密对象将失败。
- 安全影响：该政策通过对所有操作强制执行密钥承诺来提供最高级别的安全性。只有在使用密钥承诺对所有对象进行重新加密并且所有客户端都已升级到 V3 之后，才使用此策略。
- 版本兼容性：使用此策略加密的对象只能由 V3 和 S3 加密客户端的最新 V2 实现进行读取。此策略还禁止读取由 V2 或 V1 客户端加密的对象。

Note

在规划迁移时，首先 `REQUIRE_ENCRYPT_ALLOW_DECRYPT` 要保持向后兼容性，同时获得对新对象的密钥承诺所带来的安全优势。只有 `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` 在所有对象都重新加密并且所有客户端都已升级到 V3 之后才会移至。

带有关键承诺的 AES GCM

带有密钥承诺的 AES GCM (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) 是 V3 中引入的一种新加密算法，它通过防止数据密钥被篡改来增强安全性。了解此算法的工作原理以及何时适用，对于规划迁移非常重要。

与以前的算法有何 `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 不同

S3 加密客户端的先前版本使用 AES CBC 或 AES GCM，但没有密钥承诺，对指令文件中的数据密钥进行加密。`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 向加密过程添加加密承诺，将加密数据绑定到特定密钥。这样可以防止攻击者篡改指令文件中的加密数据密钥并导致客户端使用错误的密钥解密数据。

如果没有密钥承诺，攻击者就有可能修改指令文件中的加密数据密钥，使其解密为不同的密钥，从而可能允许未经授权的访问或数据损坏。`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 通过确保加密的数据密钥只能解密为加密期间使用的原始密钥来防止这种攻击。

版本兼容性

使用加密的对象 `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` 只能通过 S3 加密客户端的 V3 实现以及支持读取 V3 格式的 V2 的某些过渡版本进行解密。没有此过渡支持的 V2 客户端无法解密使用加密的指令文件。`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

Warning

在启用加密之前 `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (通过使用 `REQUIRE_ENCRYPT_ALLOW_DECRYPT` 或 `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` 承诺策略)，请确保所有需要读取您的加密对象的客户端都已升级到 V3 或支持 V3 格式的过渡版本。如果任何不支持过渡的 V2 客户端尝试读取使用加密的对象 `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`，则解密将失败。

在迁移过程中，您可以使用FORBID_ENCRYPT_ALLOW_DECRYPT承诺策略继续加密，而不必ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY同时允许 V3 客户端读取使用密钥承诺加密的对象。这提供了一个安全的迁移路径，您可以先升级所有读取器，然后切换到使用密钥承诺进行加密。

更新现有客户端以读取新格式

V3 加密客户端使用 V2 客户端默认不支持的加密算法和密钥承诺功能。迁移的第一步是将 V2 解密客户端更新为可以读取 V3 加密对象的 AWS SDK for Ruby 版本。完成此步骤后，您的应用程序的 V2 客户端将能够解密由 V3 加密客户端加密的对象。

要读取由 V3 客户端（使

用REQUIRE_ENCRYPT_ALLOW_DECRYPT或REQUIRE_ENCRYPT_REQUIRE_DECRYPT承诺策略的客户端）加密的对象，您需要使用 Gem 版本 1.93.0 或更高版本。aws-sdk-s3此版本支持使用密钥承诺解密使用 AES GCM 加密的对象。

从命令行安装

对于通过命令行安装 aws-sdk-s3 gem 的项目，请使用版本选项验证是否安装了最低版本 1.208.0。

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

使用 Gemfile

对于使用 Gemfile 管理依赖关系的项目，请将 G aws-sdk-s3 em 的最低版本设置为 1.208.0。例如：

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. 修改您的 Gemfile 以指定最低版本。
2. 运行bundle update aws-sdk-s3更新 gem。
3. 要验证您的版本，请运行 bundle info aws-sdk-s3。

Note

更新到最新版本后，您现有的 V2 加密客户端将能够解密由 V3 客户端加密的对象。但是，他们将继续使用 V2 算法对新对象进行加密，直到您按照下一节所述将其迁移到 V3。

将加密和解密客户端迁移到 V3

更新客户端以读取新的加密格式后，您可以将应用程序更新为 V3 加密和解密客户端。以下步骤向您展示了如何成功地将代码从 V2 迁移到 V3。

在更新代码以使用 V3 加密客户端之前，请确保您已按照上述步骤操作并使用的是 `aws-sdk-s3` Gem 版本 1.93.0 或更高版本。

Note

使用 AES-GCM 解密时，在开始使用解密的数据之前，请通读整个对象。这是为了验证自加密以来是否未对对象进行过修改。

配置 V3 客户端

V3 加密客户端引入了新的配置选项，用于控制密钥承诺行为和向后兼容性。了解这些选项对于成功迁移至关重要。

承诺政策

该 `commitment_policy` 参数控制加密客户端在加密和解密操作期间如何处理密钥承诺。这是 V3 客户端最重要的配置选项。

- `:require_encrypt_allow_decrypt`-使用密钥承诺对新对象进行加密，并允许使用或不使用密钥承诺解密对象。这是迁移的推荐设置，因为它可以增强新对象的安全性，同时保持与现有 V2 对象的向后兼容性。
- `:forbid_encrypt_allow_decrypt`-无需密钥承诺即可加密新对象（使用 V2 算法），并允许使用或不使用密钥承诺解密对象。只有在迁移期间需要保持 V2 加密行为时，例如某些客户端还无法读取 V3 加密对象时，才使用此设置。
- `:require_encrypt_require_decrypt`-使用密钥承诺加密新对象，并且仅允许解密使用密钥承诺加密的对象。只有在使用密钥承诺对所有对象进行重新加密并且所有客户端都已升级到 V3 之后，才使用此设置。

安全配置文件

该 `security_profile` 参数决定是否支持读取由较早版本的加密客户端写入的对象。此参数对于在迁移期间保持向后兼容性至关重要。

- `:v3_and_legacy`-允许 V3 客户端解密由 V1 和 V2 加密客户端加密的对象。在迁移期间使用此设置可确保您的 V3 客户端可以读取所有现有的加密对象。
- `:v3`-仅允许 V3 客户端解密由 V2 加密客户端加密的对象。如果您已经将所有 V1 对象迁移为 V2 格式，请使用此设置。
- 如果未指定，则客户端将仅解密由 V3 客户端加密的对象。仅将其用于不存在旧对象的新应用程序开发。

信封_位置

该 `envelope_location` 参数决定加密元数据（包括加密的数据密钥）的存储位置。此参数影响哪些对象受 AES GCM 的保护，并使用密钥承诺。

- `:metadata`（默认）-将加密元数据存储于 S3 对象的元数据标头中。这是默认行为，建议在大多数用例中使用。使用元数据存储时，带有密钥承诺的 AES GCM 不适用。
- `:instruction_file`-将加密元数据存储于具有可配置后缀的单独的 S3 对象（指令文件）中。使用说明文件时，带有密钥承诺的 AES GCM 可保护加密的数据密钥免遭篡改。如果您需要密钥承诺为数据密钥本身提供的额外安全性，请使用此设置。

使用时：`instruction_file`，您可以选择指定 `instruction_file_suffix` 参数以自定义用于指令文件对象的后缀。默认后缀为 `.instruction`

何时使用每个配置选项

在迁移过程中，请遵循以下推荐的配置策略：

1. 初始迁移：设

置 `commitment_policy: :require_encrypt_allow_decrypt` 和 `security_profile: :v3_and_legacy`。这允许您的 V3 客户端使用密钥承诺加密新对象，同时仍然能够解密所有现有的 V1 和 V2 对象。

2. 升级所有客户端后：继续使

用 `commitment_policy: :require_encrypt_allow_decrypt` 和 `security_profile: :v3_and_legacy`，直到重新加密所有需要密钥承诺保护的對象。

3. 完整 V3 强制执行：只有在使用密钥承诺重新加密所有对象并且您不再需要读取 V1/V2 对象之后，您才可以选择切换到 `commitment_policy: :require_encrypt_require_decrypt` 并删除该 `security_profile` 参数（或者 `:v2` 如果 V2 对象仍然存在，则将其设置为 `:v2`）。

对于 `envelope_location`，请继续使用现有的存储方法（`:metadata`或`:instruction_file`），除非您有具体的理由要对其进行更改。如果您当前正在使用元数据存储，并且希望获得带有密钥承诺的 AES GCM 的额外安全性，则可以切换到 `:instruction_file`，但请注意，这将需要更新所有读取这些对象的客户端。

将加密和解密客户端迁移到 V3

更新客户端以读取新的加密格式后，您可以将应用程序更新为 V3 加密和解密客户端。以下示例向您展示了如何成功地将代码从 V2 迁移到 V3。

使用 V3 加密客户端

迁移前 (V2)

```
require 'aws-sdk-s3'

# Create V2 encryption client with KMS
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy,
  commitment_policy: :forbid_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

迁移期间 (具有向后兼容性的 V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
```

```
    commitment_policy: :require_encrypt_allow_decrypt
  )

  # Encrypt and upload object
  client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

  # Download and decrypt object
  resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
  decrypted_data = resp.body.read
```

迁移后 (V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3,
  # Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
  # This encrypts with key commitment and does not decrypt V2 objects
  commitment_policy: :require_encrypt_require_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

V3 的主要区别在于 `commitment_policy` 参数的添加。将其设置为 `:require_encrypt_require_decrypt` 可确保使用密钥承诺对新对象进行加密，并且客户端仅解密使用密钥承诺加密的对象，从而增强安全性，防止数据密钥被篡改。

通常 `put_object` 话本身保持不变。所有安全增强功能都是在客户端级别配置的。

其他示例

本节提供了有关特定迁移场景和配置选项的其他示例，这些示例可能在 V2 到 V3 迁移期间很有用。

指令文件与元数据存储

S3 加密客户端可以将加密元数据（包括加密的数据密钥）存储在两个不同的位置：在 S3 对象的元数据标头中或单独的指令文件中。存储方法的选择会影响哪些对象受益于具有密钥承诺保护的 AES GCM。

元数据存储（默认）

默认情况下，加密客户端将加密元数据存储于 S3 对象的元数据标头中。这是大多数用例的推荐方法，因为它会将加密元数据与对象一起保存，并且不需要管理单独的指令文件对象。

```
require 'aws-sdk-s3'

# Create V3 encryption client with metadata storage (default)
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :metadata # Explicitly set to metadata (this is the default)
)

# Encrypt and upload object
# Encryption metadata is stored in the object's metadata headers
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')
```

使用元数据存储时，带有密钥承诺的 AES GCM 不适用于加密的数据密钥。但是，使用 `commitment_policy: :require_encrypt_allow_decrypt` 或 `require_encrypt_require_decrypt` 时，内容加密仍然受益于密钥承诺。

指令文件存储

或者，您可以将加密客户端配置为将加密元数据存储于名为指令文件的单独的 S3 对象中。在 V3 中使用说明文件时，加密的数据密钥受带有密钥承诺的 AES GCM 保护，从而提供额外的安全性，防止数据密钥被篡改。

```
require 'aws-sdk-s3'

# Create V3 encryption client with instruction file storage
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
```

```
key_wrap_schema: :kms_context,
content_encryption_schema: :aes_gcm_no_padding,
security_profile: :v3_and_legacy,
commitment_policy: :require_encrypt_allow_decrypt,
envelope_location: :instruction_file, # Store metadata in separate instruction file
instruction_file_suffix: '.instruction' # Optional: customize the suffix (default is
'.instruction')
)

# Encrypt and upload object
# Encryption metadata is stored in a separate object: 'my-object.instruction'
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# When retrieving the object, the client automatically reads the instruction file
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

使用时`envelope_location: :instruction_file`，加密客户端会创建两个 S3 对象：

1. 加密的数据对象（例如`my-object`）
2. 包含加密元数据的指令文件（例如`my-object.instruction`）

该`instruction_file_suffix`参数允许您自定义用于指令文件的后缀。默认值为`.instruction`。

何时使用每种存储方法

- 在大多数场景中@@ 使用元数据存储。它简化了对象管理，因为加密元数据会随对象一起传输。
- 当需要考虑对象元数据大小或需要将加密元数据与加密对象分开时，请@@ 使用指令文件存储。请注意，使用指令文件需要管理两个 S3 对象（加密对象及其指令文件），而不是一个。

Warning

如果从元数据存储更改为指令文件存储（反之亦然），则使用新存储方法配置的客户机将无法读取使用旧存储方法加密的现有对象。仔细规划存储方法并保持应用程序间的一致性。

文档历史记录

下表介绍了本指南的重要更改。如需获取对此文档的更新的通知，您可以订阅 [RSS 源](#)。

变更	说明	日期
内容重组	更新目录和内容组织以更好地与其他内容保持一致 AWS SDKs。	2025 年 3 月 29 日
可观察性	添加了有关 Ruby 可观测性的信息。	2025 年 1 月 24 日
常规更新	将所需的最低 Ruby 版本更新至 v2.5。更新了资源链接。	2024 年 11 月 13 日
目录与引导式示例	删除了引导式示例，相关内容请参阅更全面的代码示例存储库。	2024 年 7 月 10 日
目录	更新了目录，使代码示例更易于访问。	2023 年 6 月 1 日
IAM 最佳实践更新	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践 。入门更新。	2023 年 5 月 8 日
常规更新	更新了相关外部资源的欢迎页面。还更新了 v2.3 要求使用的最低 Ruby 版本。更新了 AWS Key Management Service 章节以反映术语更新。更新了有关 REPL 实用程序的使用信息，使其更清晰。	2022 年 8 月 8 日
更正失效链接	修复了失效的示例链接。删除了多余的“提示和技巧”页面；	2022 年 8 月 3 日

重定向到 Amazon EC2 示例内容。包含代码示例存储库 GitHub 中可用的代码示例列表。

[开发工具包指标](#)

删除了有关启用 Enterprise Support 的 SDK 指标的信息，这些信息已弃用。

2022 年 1 月 28 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。