開發人員指南

AWS 適用於 Ruby 的 SDK

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

AWS 適用於 Ruby 的 SDK: 開發人員指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是適用於 Ruby 的 AWS SDK？ 1
其他文件與資源 1
部署至 AWS 雲端 1
開發套件主要版本的維護與支援 2
開始使用 3
使用 驗證 AWS 3
使用主控台登入資料 3
使用 IAM Identity Center 身分驗證 3
更多身分驗證資訊 5
安裝 開發套件 5
先決條件 5
安裝 開發套件 6
建立簡單的應用程式 7
編寫程式碼 7
執行程式 8
Windows 使用者的注意事項 8
後續步驟 9
設定服務用戶端 10
設定的優先順序 11
外部用戶端組態 11
AWS 適用於 Ruby 的 SDK 環境變數 12
程式碼中的用戶端組態 12
Aws.config 12
AWS 區域 13
解析的區域搜尋順序 14
如何設定 區域 14
登入資料提供者 15
登入資料提供者鏈結 15
建立 AWS STS 存取字符 17
重試 18
在程式碼中指定用戶端重試行為 18
可觀測性 18
OTelProvider 為服務用戶端設定 19
OTelProvider 為所有服務用戶端設定 21

設定自訂遙測供應商 22
跨度屬性 22
HTTP 24
設定非標準端點 24
使用開發套件 25
提出 AWS 服務 請求 25
使用 REPL 公用程式 26
先決條件 26
Bundler 設定 26
執行 REPL 27
搭配 Ruby on Rails 使用 SDK 27
使用來自用戶端的線路追蹤進行偵錯 27
使用虛設測試 29
繪製用戶端回應 29
繪製用戶端錯誤 30
分頁 30
分頁回應可列舉 31
手動處理分頁回應 31
分頁資料類別 31
等待程式 32
叫用等待程式 32
等待失敗 32
設定等待程式 33
擴展等待程式 33
程式碼範例 35
Aurora 36
開始使用 36
Auto Scaling 37
開始使用 36
CloudTrail 39
動作 39
CloudWatch 43
動作 39
Amazon Cognito 身分提供者 55
開始使用 36
Amazon Comprehend 57

案例 57
Amazon DocumentDB 58
無伺服器範例 58
DynamoDB 59
開始使用 36
基本概念 61
動作 39
案例 57
無伺服器範例 58
Amazon EC2 88
開始使用 36
動作 39
Elastic Beanstalk 122
動作 39
EventBridge 128
案例 57
AWS Glue 146
開始使用 36
基本概念 61
動作 39
IAM 173
開始使用 36
基本概念 61
動作 39
Kinesis 230
無伺服器範例 58
AWS KMS 233
動作 39
Lambda 237
開始使用 36
基本概念 61
動作 39
案例 57
無伺服器範例 58
Amazon MSK 265
無伺服器範例 58

Amazon Polly 266
動作 39
案例 57
Amazon RDS 271
開始使用 36
動作 39
無伺服器範例 58
Amazon S3 279
開始使用 36
基本概念 61
動作 39
案例 57
無伺服器範例 58
Amazon SES 311
動作 39
Amazon SES API v2 316
動作 39
Amazon SNS 317
動作 39
無伺服器範例 58
Amazon SQS 327
動作 39
無伺服器範例 58
AWS STS 340
動作 39
Amazon Textract 341
案例 57
Amazon Translate 342
案例 57
遷移版本 344
Side-by-side用量 344
一般差異 344
用戶端差異 345
資源差異 346
安全 347
資料保護 347

身分和存取權管理 348
合規驗證 348
恢復能力 349
基礎設施安全性 350
強制執行最低 TLS 版本 350
檢查 OpenSSL 版本 350
升級 TLS 支援 351
S3 加密用戶端遷移 (V1 至 V2) 351
遷移概觀 351
更新現有用戶端以讀取新格式 351
將加密和解密用戶端遷移至 V2 353
S3 加密用戶端遷移 (V2 到 V3) 356
遷移概觀 356
了解 V3 功能 356
更新現有用戶端以讀取新格式 359
將加密和解密用戶端遷移至 V3 360
文件歷史記錄 366
..... ccclxviii

什麼是適用於 Ruby 的 AWS SDK？

歡迎使用適用於 Ruby 的 AWS SDK 開發人員指南。適用於 Ruby 的 AWS SDK 提供幾乎所有 的支援程式庫 AWS 服務，包括 Amazon Simple Storage Service (Amazon S3)、Amazon Elastic Compute Cloud (Amazon EC2) 和 Amazon DynamoDB。

適用於 Ruby 的 AWS SDK 開發人員指南提供有關如何安裝、設定和使用適用於 Ruby 的 AWS SDK 來建立使用 之 Ruby 應用程式的相關資訊 AWS 服務。

適用於 Ruby 的 AWS 開發套件入門

其他文件與資源

如需適用於 Ruby 開發人員的 AWS SDK 資源，請參閱下列內容：

- AWS SDKs和工具參考指南 – 包含設定、功能和其他在 AWS SDKs之間常見的基本概念
- 適用於 Ruby 的 AWS SDK API 參考 - 第 3 版
- GitHub 上的AWS 程式碼範例儲存庫
- RubyGems.org – 最新版本的 SDK 已模組化為服務特定的 Gem 套件，可在此處取得
  - 支援的服務 – 列出適用於 Ruby 的 AWS SDK 支援的所有 Gem
- AWS GitHub 上的適用於 Ruby 的 SDK 來源：
  - 來源和讀取
  - 變更每個 Gem 套件下的日誌
  - 從 v2 移至 v3
  - 問題
  - 核心升級備註
- 開發人員部落格

部署至 AWS 雲端

您可以使用 AWS 服務 和 AWS Elastic Beanstalk AWS CodeDeploy 等，將您的應用程式部署到 AWS 雲端。如需使用 Elastic Beanstalk 部署 Ruby 應用程式，請參閱《 AWS Elastic Beanstalk 開發人員指南》中的使用 EB CLI 和 Git 在 Ruby 中部署 Elastic Beanstalk 應用程式。如需 AWS 部署服務的概觀，請參閱 上的部署選項概觀 AWS。

開發套件主要版本的維護與支援

如需開發套件主要版本及其基礎相依性之維護與支援的相關資訊，請參閱《AWS 開發套件及工具參考指南》中的以下內容：

- AWS SDKs和工具維護政策
- AWS SDKs和工具版本支援矩陣

適用於 Ruby 的 AWS 開發套件入門

了解如何安裝、設定和使用 SDK 來建立 Ruby 應用程式，以程式設計方式存取 AWS 資源。

主題

- AWS 使用適用於 Ruby 的 AWS SDK 透過 驗證
- 安裝適用於 Ruby 的 AWS SDK
- 使用適用於 Ruby 的 AWS SDK 建立簡單的應用程式

AWS 使用適用於 Ruby 的 AWS SDK 透過 驗證

使用 進行開發 AWS 時，您必須建立程式碼向 進行身分驗證的方式 AWS 服務。您可以根據環境和您可用的存取權，以不同的方式設定 AWS 資源的程式設計 AWS 存取。

若要選擇您的身分驗證方法並針對 SDK 進行設定，請參閱 AWS SDKs和工具參考指南中的身分驗證和存取。

使用主控台登入資料

針對本機開發，我們建議新使用者使用現有的 AWS 管理主控台登入憑證，以程式設計方式存取 AWS 服務。在瀏覽器型身分驗證之後， AWS 會產生臨時登入資料，以使用本機開發工具，例如 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Ruby。

如果您選擇此方法，請遵循使用 AWS CLI 使用主控台登入資料進行 AWS 本機開發的說明。

適用於 Ruby 的 AWS SDK 不需要將其他 Gem 套件 （ 例如 `aws-sdk-signin`) 新增至您的應用程式，即可搭配主控台登入資料使用登入。

使用 IAM Identity Center 身分驗證

如果您選擇此方法，請完成 AWS SDKs 和工具參考指南中的 IAM Identity Center 身分驗證程序。之後，您的環境應該包含下列元素：

- 在執行應用程式之前 AWS CLI，您用來啟動 AWS 存取入口網站工作階段的 。
- 共用AWSconfig檔案，其`[default]`設定檔具有一組可從 SDK 參考的組態值。若要尋找此檔案的位置，請參閱 AWS SDK 和工具參考指南中的共用檔案位置。
- 共用`config`檔案會設定 `region`設定。這會設定軟體開發套件用於 AWS 請求 AWS 區域 的預設值。此區域用於未指定使用 區域的 SDK 服務請求。

- 開發套件使用描述檔的 SSO 字符提供者組態，在傳送請求至 之前取得登入資料 AWS。此sso_role_name值是連接至 IAM Identity Center 許可集的 IAM 角色，允許存取您應用程式中 AWS 服務 使用的 。

  下列範例config檔案顯示使用 SSO 字符提供者組態設定的預設設定檔。設定檔sso_session的設定是指具名sso-session區段。sso-session 區段包含啟動 AWS 存取入口網站工作階段的設定。

```
[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
```

適用於 Ruby 的 AWS SDK 不需要將其他 Gem 套件 （例如 aws-sdk-sso和 aws-sdk-ssooidc) 新增至您的應用程式，即可使用 IAM Identity Center 身分驗證。

啟動 AWS 存取入口網站工作階段

在執行存取的應用程式之前 AWS 服務，您需要 SDK 的作用中 AWS 存取入口網站工作階段，才能使用 IAM Identity Center 身分驗證來解析登入資料。視您設定的工作階段長度而定，您的存取最終會過期，而且 SDK 會遇到身分驗證錯誤。若要登入 AWS 存取入口網站，請在 中執行下列命令 AWS CLI。

```
aws sso login
```

如果您遵循指引並設定預設設定檔，則不需要使用 --profile選項呼叫 命令。如果您的 SSO 權杖提供者組態使用已命名的設定檔，則命令為 aws sso login --profile named-profile。

若要選擇性地測試您是否已經有作用中的工作階段，請執行下列 AWS CLI 命令。

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

如果您的工作階段處於作用中狀態，對此命令的回應會報告共用`config`檔案中設定的 IAM Identity Center 帳戶和許可集。

> **Note**
>
> 如果您已有作用中的 AWS 存取入口網站工作階段並執行 `aws sso login`，則不需要提供登入資料。
>
> 登入程序可能會提示您允許 AWS CLI 存取您的資料。由於 AWS CLI 建置在適用於 Python 的 SDK 之上，因此許可訊息可能包含`botocore`名稱的變化。

更多身分驗證資訊

人類使用者具有人類身分，是應用程式的相關人員、管理員、開發人員、操作員和消費者。它們必須具有身分才能存取您的 AWS 環境和應用程式。屬於您組織成員的人類使用者 - 這表示身為開發人員的您 - 稱為人力身分。

存取時使用臨時登入資料 AWS。您可以為您的人類使用者使用身分提供者，透過擔任提供臨時登入資料的角色來提供 AWS 帳戶的聯合存取。對於集中式存取管理，我們建議您使用 AWS IAM Identity Center (IAM Identity Center) 來管理對帳戶和這些帳戶中許可的存取。如需更多替代方案，請參閱下列內容：

- 如需了解有關最佳實務的資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。
- 若要建立短期 AWS 登入資料，請參閱《IAM 使用者指南》中的暫時安全登入資料。
- 若要了解適用於 Ruby 的 AWS SDK 登入資料提供者鏈結，以及 SDK 如何依序自動嘗試不同的身分驗證方法，請參閱 登入資料提供者鏈結。
- 如需 AWS SDK 登入資料提供者組態設定，請參閱 SDK AWS SDKs 和工具參考指南中的標準化登入資料提供者。

安裝適用於 Ruby 的 AWS SDK

本節包含適用於 Ruby 的 AWS SDK 的先決條件和安裝指示。

先決條件

在使用適用於 Ruby 的 AWS SDK 之前，您必須先向 進行身分驗證 AWS。如需設定身分驗證的資訊，請參閱 AWS 使用適用於 Ruby 的 AWS SDK 透過 驗證。

安裝 開發套件

您可以像安裝 Ruby Gem 套件一樣安裝適用於 Ruby 的 AWS SDK。Gem 套件可在 RubyGems 取得。適用於 Ruby 的 AWS SDK 設計為模組化，並以 分隔 AWS 服務。安裝整個 `aws-sdk` Gem 套件很大，可能需要一個小時的時間。

建議您只為您 AWS 服務 使用的 安裝 Gem 套件。這些名稱命名為 ，`aws-sdk-`*`service_abbreviation`*完整清單位於適用於 Ruby README 的 AWS SDK 檔案的支援服務資料表中。例如，與 Amazon S3 服務連接的 Gem 套件可直接在 取得`aws-sdk-s3`。

Ruby 版本管理員

建議使用 Ruby 版本管理工具，而不是使用系統 Ruby，如下所示：

- RVM
- 夏爾比
- rbenv

例如，如果您使用的是 Amazon Linux 2 作業系統，下列命令可用來更新 RVM、列出可用的 Ruby 版本，然後選擇您想要使用適用於 Ruby 的 AWS SDK 進行開發的版本。Ruby 最低必要版本為 2.5。

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

Bundler

如果您使用 Bundler，下列命令會安裝適用於 Amazon S3 的 Ruby Gem AWS 開發套件：

1. 安裝 Bundler 並建立 `Gemfile`：

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

2. 開啟已建立的 `Gemfile`，並為程式碼將使用的每個 AWS 服務 Gem 新增一`gem`行。若要遵循 Amazon S3 範例，請將以下行新增至檔案底部：

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

3. 儲存 Gemfile。
4. 安裝 中指定的相依性Gemfile：

```
$ bundle install
```

使用適用於 Ruby 的 AWS SDK 建立簡單的應用程式

使用適用於 Ruby 的 AWS SDK 向 Amazon S3 打招呼。下列範例顯示 Amazon S3 儲存貯體的清單。

編寫程式碼

將下列程式碼複製並貼到新的來源檔案中。將檔案命名為 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 適用於 Ruby 的 SDK 設計為模組化，並以 分隔 AWS 服務。安裝 Gem 套件後，Ruby 來源檔案頂端的 `require`陳述式會匯入 Amazon S3 服務的 AWS SDK 類別和方法。如需可用 AWS 服務 Gem 的完整清單，請參閱適用於 Ruby README 檔案的 AWS SDK 支援服務資料表。

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

執行程式

開啟命令提示以執行 Ruby 程式。執行 Ruby 程式的典型命令語法為：

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

此範例程式碼不使用引數。若要執行此程式碼，請在命令提示中輸入以下內容：

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

Windows 使用者的注意事項

當您在 Windows 上使用 SSL 憑證並執行 Ruby 程式碼時，您可能會看到類似以下的錯誤。

```
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'
...
```

若要修正此問題，請在第一次 AWS 呼叫之前某個位置，將以下行新增至 Ruby 來源檔案。

```
Aws.use_bundled_cert!
```

如果您在 Ruby 程式中僅使用 `aws-sdk-s3` Gem 套件，並且想要使用綁定憑證，則還需要新增 `aws-sdk-core` Gem 套件。

後續步驟

若要測試許多其他 Amazon S3 操作，請查看 GitHub 上的AWS 程式碼範例儲存庫。

在適用於 Ruby 的 AWS SDK 中設定服務用戶端

若要以程式設計方式存取 AWS 服務，適用於 Ruby 的 AWS SDK 會針對每個 使用用戶端類別 AWS 服務。例如，如果您的應用程式需要存取 Amazon EC2，您的應用程式會建立 Amazon EC2 用戶端物件以與該服務連接。然後，您可以使用 服務用戶端向該用戶端提出請求 AWS 服務。

若要向 提出請求 AWS 服務，您必須先建立服務用戶端。對於 AWS 服務 您的程式碼使用的每個 ，它都有自己的 Gem 套件和自己的專用類型來與其互動。用戶端會針對服務公開的每個 API 操作公開一個方法。

設定 SDK 行為有許多替代方法，但最終一切都與服務用戶端的行為有關。使用從中建立的服務用戶端之前，任何組態都不會生效。

當您使用 開發 AWS 時，您必須建立程式碼向 進行身分驗證的方式 AWS 服務。您也必須設定 AWS 區域 要使用的 。

AWS SDKs和工具參考指南也包含許多 AWS SDKs 中常見的設定、功能和其他基本概念。

主題

- 設定的優先順序
- 在外部設定適用於 Ruby 服務的 AWS SDK 服務用戶端
- 在程式碼中設定適用於 Ruby 服務用戶端的 AWS SDK
- AWS 區域 為適用於 Ruby 的 AWS SDK 設定
- 使用適用於 Ruby 的 AWS SDK 登入資料提供者
- 在適用於 Ruby 的 AWS SDK 中設定重試
- 在適用於 Ruby 的 AWS SDK 中設定可觀測性功能
- 在適用於 Ruby 的 AWS SDK 中設定 HTTP 層級設定

共用 `config`和 `credentials` 檔案可用於組態設定。如需所有 AWS SDK 設定，請參閱 SDK 和工具參考指南中的設定參考。 AWS SDKs

不同的設定檔可用來存放不同的組態。若要指定 SDK 載入的作用中設定檔，您可以使用 `AWS_PROFILE`環境變數或 `profile`的選項`Aws.config`。

設定的優先順序

全域設定可設定大多數 SDKs 支援且具有廣泛影響的功能、登入資料提供者和其他功能 AWS 服務。AWS SDKs有一系列要檢查的位置（或來源），以尋找全域設定的值。並非所有設定都可在所有來源中使用。以下是設定查詢優先順序：

1. 程式碼中或服務用戶端本身上設定的任何明確設定，都優先於任何其他設定。
   a. 直接傳遞至用戶端建構函數的任何參數都具有最高的優先順序。
   b. `Aws.config` 會檢查全域或服務特定的設定。
2. 檢查 環境變數。
3. 已檢查共用 AWS `credentials`檔案。
4. 已檢查共用 AWS `config`檔案。
5. 適用於 Ruby 的 AWS 開發套件原始碼本身所提供的任何預設值都會最後使用。

在外部設定適用於 Ruby 服務的 AWS SDK 服務用戶端

許多組態設定可以在程式碼之外處理。在外部處理組態時，組態會套用至所有應用程式。大多數組態設定可以設定為環境變數或單獨的共用 AWS `config`檔案中。共用`config`檔案可以維護個別的設定集，稱為設定檔，為不同的環境或測試提供不同的組態。

環境變數和共用`config`檔案設定會在 SDKs 和工具之間 AWS 標準化和共用，以支援不同程式設計語言和應用程式的一致功能。

請參閱 AWS SDKs和工具參考指南，了解如何透過這些方法設定您的應用程式，以及每個跨 sdk 設定的詳細資訊。若要查看開發套件可從環境變數或組態檔案解析的所有設定，請參閱開發套件和工具參考指南中的設定參考。 AWS SDKs

若要向 提出請求 AWS 服務，您必須先執行個體化該服務的用戶端。您可以設定服務用戶端的常見設定，例如逾時、HTTP 用戶端和重試組態。

每個服務用戶端都需要 AWS 區域 和登入資料提供者。SDK 使用這些值將請求傳送到 資源的正確區域，並使用正確的登入資料簽署請求。您可以在程式碼中以程式設計方式指定這些值，或從環境中自動載入這些值。

開發套件有一系列位置（或來源）可供其檢查，以尋找組態設定的值。

1. 程式碼中或服務用戶端本身上設定的任何明確設定，都優先於任何其他設定。

2. 環境變數
   - 如需設定環境變數的詳細資訊，請參閱 AWS SDKs和工具參考指南中的環境變數。
   - 請注意，您可以為殼層設定不同範圍的環境變數：全系統、全使用者和特定終端機工作階段。
3. 共用 config和 credentials 檔案
   - 如需設定這些檔案的詳細資訊，請參閱 SDK config和工具參考指南中的共用 和 credentials 檔案。 AWS SDKs
4. 開發套件原始碼本身提供的任何預設值都會最後使用。
   - 有些屬性，例如區域，沒有預設值。您必須在程式碼、環境設定或共用config檔案中明確指定它們。如果 SDK 無法解析必要的組態，API 請求可能會在執行時間失敗。

AWS 適用於 Ruby 的 SDK 環境變數

除了大多數 AWS SDKs支援的跨磁碟環境變數之外，適用於 Ruby 的 AWS SDK 還支援一些獨特的變數：

`AWS_SDK_CONFIG_OPT_OUT`

如果AWS_SDK_CONFIG_OPT_OUT已設定適用於 Ruby 的 AWS SDK 環境變數，則共用 AWS config檔案通常 ~/.aws/config不會用於任何組態值。

`AMAZON_REGION`

的替代環境變數，AWS_REGION用於設定 AWS 區域。只有在AWS_REGION未使用 時，才會檢查此值。

在程式碼中設定適用於 Ruby 服務用戶端的 AWS SDK

直接在程式碼中處理組態時，組態範圍僅限於使用該程式碼的應用程式。在該應用程式中，有所有服務用戶端的全域組態、特定 AWS 服務 類型之所有用戶端的組態，或特定服務用戶端執行個體的組態等選項。

Aws.config

若要在所有 AWS 類別的程式碼中提供全域組態，請使用 aws-sdk-core Gem Aws.config 套件中提供的 。

Aws.config 支援兩種不同用途的語法。全域設定可以套用至所有 AWS 服務 或特定服務。如需支援設定的完整清單，請參閱 適用於 Ruby 的 AWS SDK API 參考ClientOptions中的 。

透過 進行全域設定 `Aws.config`

若要透過 設定服務無關的設定`Aws.config`，請使用下列語法：

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

這些設定會合併到任何建立的服務用戶端。

全域設定的範例：

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

如果您嘗試使用全域不支援的設定名稱，當您嘗試建立不支援該名稱的服務類型的執行個體時，就會引發錯誤。如果發生這種情況，請改用服務特定的語法。

透過 的服務特定設定 `Aws.config`

若要透過 設定服務特定的設定`Aws.config`，請使用下列語法：

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

這些設定會合併到該服務類型的所有已建立服務用戶端。

僅適用於 Amazon S3 的設定範例：

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

<service identifier> 可以透過查看AWS 適用於 Ruby Gem 的對應 SDK 名稱，以及使用後面的尾碼 "aws-sdk-" 來識別 。例如：

- 對於 aws-sdk-s3，服務識別符字串為 "s3"。
- 對於 aws-sdk-ecs，服務識別符字串為 "ecs"。

AWS 區域 為適用於 Ruby 的 AWS SDK 設定

您可以使用 存取在特定地理區域中操作 AWS 服務 的 AWS 區域。這對於備援和讓資料和應用程式靠近您和您的使用者存取它們的位置，都很有用。

> **Important**
>
> 大多數資源都位於特定 中， AWS 區域 您必須在使用 SDK 時為資源提供正確的區域。

您必須為適用於 Ruby AWS 區域 的 SDK 設定預設值，以用於 AWS 請求。此預設值用於非以 區域指定的任何 SDK 服務方法呼叫。

如需 `region`設定的詳細資訊，請參閱《 AWS SDKs與工具參考指南AWS 區域》中的 。這也包含如何透過共用 AWS `config`檔案或環境變數設定預設區域的範例。

解析的區域搜尋順序

您需要在使用大多數 時設定區域 AWS 服務。適用於 Ruby 的 AWS SDK 會依下列順序搜尋區域：

1. 在用戶端或資源物件中設定區域
2. 使用 設定區域 `Aws.config`
3. 使用環境變數設定區域
4. 使用共用`config`檔案設定區域

如何設定 區域

本節說明設定區域的不同方式，從最常見的方法開始。

使用共用`config`檔案設定區域

透過在共用 AWS `config`檔案中設定 `region`變數來設定區域。如需共用`config`檔案的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的共用組態和登入資料檔案。

在 `config` 檔案中設定此值的範例：

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

如果`AWS_SDK_CONFIG_OPT_OUT`已設定環境變數，則不會檢查共用`config`檔案。

使用環境變數設定區域

透過設定 `AWS_REGION`環境變數來設定 區域。

使用 export命令在以 Unix 為基礎的系統上設定此變數，例如 Linux 或 macOS。下列範例會將 區域設定為 us-west-2。

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

若要在 Windows 上設定此變數，請使用 set命令。下列範例會將 區域設定為 us-west-2。

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

使用 設定區域 Aws.config

將region值新增至Aws.config雜湊來設定區域。下列範例會將Aws.config雜湊更新為使用 us-west-1區域。

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

您稍後建立的任何用戶端或資源都會繫結至此區域。

在用戶端或資源物件中設定區域

在建立 AWS 用戶端或資源時設定 區域。下列範例會在 us-west-1區域中建立 Amazon S3 資源物件。為您的 AWS 資源選擇正確的區域。服務用戶端物件不可變，因此您必須為您提出請求的每個服務建立新的用戶端，並使用不同的組態對相同的服務提出請求。

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

使用適用於 Ruby 的 AWS SDK 登入資料提供者

所有對 的請求 AWS 都必須使用 發行的登入資料以密碼編譯方式簽署 AWS。在執行時間，軟體開發套件會檢查多個位置，以擷取登入資料的組態值。

使用 進行身分驗證 AWS 可以在您的程式碼庫之外處理。開發套件可以使用登入資料提供者鏈自動偵測、使用和重新整理許多身分驗證方法。

如需專案身分 AWS 驗證入門的引導選項，請參閱AWS SDKs和工具參考指南》中的身分驗證和存取。

登入資料提供者鏈結

所有 SDKs 都有一系列位置 （或來源） 進行檢查，以取得可用於向 提出請求的有效登入資料 AWS 服務。找到有效的憑證後，系統就會停止搜尋。此系統搜尋稱為預設登入資料提供者鏈結。

> **Note**
>
> 如果您遵循建議的方法讓新使用者開始使用，您會在 期間使用登入與主控台登入資料進行身分驗證AWS 使用適用於 Ruby 的 AWS SDK 透過 驗證。其他身分驗證方法適用於不同的情況。為了避免安全風險，我們建議您一律使用短期登入資料。如需其他身分驗證方法程序，請參閱AWS SDKs和工具參考指南》中的身分驗證和存取。

對於鏈結中的每個步驟，有不同的方法來設定值。直接在程式碼中設定值一律優先，接著設定為環境變數，然後在共用 AWS `config`檔案中設定。

AWS SDKs和工具參考指南提供 AWS SDKs和 所使用的開發套件組態設定資訊 AWS CLI。若要進一步了解如何透過共用 AWS `config`檔案設定 SDK，請參閱共用組態和登入資料檔案。若要進一步了解如何透過設定環境變數來設定 SDK，請參閱環境變數支援。

若要進行身分驗證 AWS，適用於 Ruby 的 AWS SDK 會依照下表所列的順序檢查登入資料提供者。

登入資料提供者優先順序	AWS SDKs與工具參考指南	適用於 Ruby 的 AWS SDK API 參考
AWS 存取金鑰 （暫時和長期登入資料）	AWS 存取金鑰	`Aws::Credentials` `Aws::SharedCredentials`
Web 身分字符來自 AWS Security Token Service (AWS STS)	擔任角色登入資料提供者 使用 `role_arn`、`role_session_name` 和 `web_identity_token_file`	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center。 在本指南中，請參閱 AWS 使用適用於 Ruby 的 AWS SDK 透過 驗證。	IAM Identity Center 憑證提供者	`Aws::SSOCredentials`
信任的實體提供者 （例如 `AWS_ROLE_ARN` ）。在本指南	擔任角色登入資料提供者	`Aws::AssumeRoleCredentials`

登入資料提供者優先順序	AWS SDKs與工具參考指南	適用於 Ruby 的 AWS SDK API 參考
中，請參閱 建立 AWS STS 存取字符。	使用 role_arn和 role_session_name	
登入憑證提供者	登入憑證提供者	Aws::LoginCredentials
程序登入資料提供者	程序登入資料提供者	Aws::ProcessCredentials
Amazon Elastic Container Service (Amazon ECS) 登入資料	容器憑證提供者	Aws::ECSCredentials
Amazon Elastic Compute Cloud (Amazon EC2) 執行個體設定檔登入資料 (IMDS 登入資料提供者)	IMDS 登入資料提供者	Aws::InstanceProfileCredentials

如果AWS_SDK_CONFIG_OPT_OUT已設定適用於 Ruby 的 AWS SDK 環境變數，則通常在 的共用 AWS config檔案 ~/.aws/config將不會剖析登入資料。

建立 AWS STS 存取字符

假設角色涉及使用一組臨時安全登入資料，您可以使用這些登入資料來存取您通常無法存取 AWS 的資源。這些臨時登入資料由存取金鑰 ID、私密存取金鑰和安全字符組成。您可以使用 Aws::AssumeRoleCredentials方法來建立 AWS Security Token Service (AWS STS) 存取字符。

下列範例使用存取字符來建立 Amazon S3 用戶端物件，其中 linked::account::arn是要擔任角色的 Amazon Resource Name (ARN)， session-name是擔任角色工作階段的識別符。

```
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)
```

如需有關設定 `role_arn`或 `role_session_name`，或改用共用 AWS `config`檔案設定這些項目的詳細資訊，請參閱 AWS SDKs和工具參考指南中的擔任角色登入資料提供者。

在適用於 Ruby 的 AWS SDK 中設定重試

適用於 Ruby 的 AWS SDK 為服務請求和可自訂的組態選項提供預設的重試行為。呼叫 AWS 服務 偶爾會傳回未預期的例外狀況。如果重試呼叫，某些類型的錯誤可能會成功，例如限流或暫時性錯誤。

您可以使用共用 AWS `config`檔案中的環境變數或設定，全域設定重試行為。如需此方法的資訊，請參閱 AWS SDKs和工具參考指南中的重試行為。它也包含重試策略實作的詳細資訊，以及如何逐一選擇。

或者，您也可以在程式碼中設定這些選項，如下列各節所示。

在程式碼中指定用戶端重試行為

根據預設，適用於 Ruby 的 AWS SDK 最多會執行三次重試，兩次重試之間間隔 15 秒，總共最多四次嘗試。因此， 操作最多可能需要 60 秒才能逾時。

下列範例會在區域 中建立 Amazon S3 用戶端`us-west-2`，並指定 在每個用戶端操作的兩次重試之間等待五秒。因此，Amazon S3 用戶端操作最多可能需要 15 秒才能逾時。

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

程式碼中或服務用戶端本身上設定的任何明確設定，都優先於環境變數或共用`config`檔案中的設定。

在適用於 Ruby 的 AWS SDK 中設定可觀測性功能

可觀測性是從系統發出的資料推斷系統目前狀態的程度。發出的資料通常稱為遙測。適用於 Ruby 的 AWS SDK 可以提供追蹤作為遙測訊號。您可以連接 `TelemetryProvider`以收集遙測資料並將其傳送至可觀測性後端。開發套件目前支援 OpenTelemetry (OTel) 做為遙測供應商，而 OpenTelemetry 有許多匯出遙測資料的方式，包括使用 AWS X-Ray或 Amazon CloudWatch。如需適用於 Ruby 的 OpenTelemetry 匯出工具的詳細資訊，請參閱 OpenTelemetry 網站上的匯出工具。

根據預設，軟體開發套件不會記錄或發出任何遙測資料。本主題說明如何設定和發出遙測輸出。

您可以針對特定服務或全域設定遙測。適用於 Ruby 的 SDK 提供 OpenTelemetry 供應商。您也可以定義您選擇的自訂遙測供應商。

`OTelProvider` 為服務用戶端設定

適用於 Ruby 的 SDK 提供稱為 的 OpenTelemetry 供應商`OTelProvider`。下列範例使用 Amazon Simple Storage Service 服務用戶端的 OpenTelemetry 設定遙測匯出。在此簡單範例中，執行程式碼時，來自 OpenTelemetry `OTEL_TRACES_EXPORTER`的環境變數會用來將追蹤匯出至主控台輸出。若要進一步了解 `OTEL_TRACES_EXPORTER`，請參閱 OpenTelemetry 文件中的匯出工具選擇。

```
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
```

先前的程式碼範例顯示為服務用戶端設定追蹤輸出的步驟：

1. 需要 OpenTelemetry 相依性。
   a. `opentelemetry-sdk` 用於使用 `Aws::Telemetry::OTelProvider`。
   b. `opentelemetry-exporter-otlp` 用於匯出遙測資料。
2. 呼叫 `OpenTelemetry::SDK.configure` 以使用其組態預設值設定 OpenTelemetry SDK。
3. 使用適用於 Ruby 的 OpenTelemetry 供應商的 SDK，建立 執行個體`OTelProvider`，以做為組態選項傳遞至您要追蹤的服務用戶端。

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

使用這些步驟，在該服務用戶端上呼叫的任何方法都會發出追蹤資料。

從呼叫 Amazon S3 `list_buckets`方法產生的追蹤輸出範例如下：

OpenTelemetry 追蹤輸出範例

```
#<struct OpenTelemetry::SDK::Trace::SpanData
 name="Handler.NetHttp",
 kind=:internal,
 status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
 parent_span_id="\xBFb\xC9\xFD\xA6F!\xE1",
 total_recorded_attributes=7,
 total_recorded_events=0,
 total_recorded_links=0,
 start_timestamp=1736190567061767000,
 end_timestamp=1736190567317160000,
 attributes=
  {"http.method"=>"GET",
   "net.protocol.name"=>"http",
   "net.protocol.version"=>"1.1",
   "net.peer.name"=>"s3.amazonaws.com",
   "net.peer.port"=>"443",
   "http.status_code"=>"200",
   "aws.request_id"=>"22HSH7NQTYMB5NHQ"},
 links=nil,
 events=nil,
 resource=
  #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
   @attributes=
    {"service.name"=>"unknown_service",
     "process.pid"=>37013,
     "process.command"=>"example.rb",
     "process.runtime.name"=>"ruby",
     "process.runtime.version"=>"3.3.0",
     "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
 [arm64-darwin23]",
     "telemetry.sdk.name"=>"opentelemetry",
     "telemetry.sdk.language"=>"ruby",
     "telemetry.sdk.version"=>"1.6.0"}>,
 instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
 name="aws.s3.client", version="">,
 span_id="\xEF%\x9C\xB5\x8C\x04\xDB\x7F",
 trace_id=" \xE7\xF1\xF8\x9D\e\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
 trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
 tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
#<struct OpenTelemetry::SDK::Trace::SpanData
 name="S3.ListBuckets",
 kind=:client,
```

```
status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
parent_span_id="\x00\x00\x00\x00\x00\x00\x00\x00",
total_recorded_attributes=5,
total_recorded_events=0,
total_recorded_links=0,
start_timestamp=1736190567054410000,
end_timestamp=1736190567327916000,
attributes={"rpc.system"=>"aws-api", "rpc.service"=>"S3", "rpc.method"=>"ListBuckets",
"code.function"=>"list_buckets", "code.namespace"=>"Aws::Plugins::Telemetry"},
links=nil,
events=nil,
resource=
 #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
  @attributes=
   {"service.name"=>"unknown_service",
    "process.pid"=>37013,
    "process.command"=>"example.rb",
    "process.runtime.name"=>"ruby",
    "process.runtime.version"=>"3.3.0",
    "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
[arm64-darwin23]",
    "telemetry.sdk.name"=>"opentelemetry",
    "telemetry.sdk.language"=>"ruby",
    "telemetry.sdk.version"=>"1.6.0"}>,
instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
name="aws.s3.client", version="">,
span_id="\xBFb\xC9\xFD\xA6F!\xE1",
trace_id=" \xE7\xF1\xF8\x9D\e\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
```

先前的追蹤輸出有兩個資料範圍。每個追蹤項目提供一或多個屬性中事件的其他中繼資料。

OTelProvider 為所有服務用戶端設定

您可以選擇全域開啟遙測，而不是開啟前一節所述的特定服務用戶端的遙測。

若要為所有 AWS 服務用戶端發出遙測資料，您可以在建立服務用戶端Aws.config之前，在 上設定遙測提供者。

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

使用此組態，之後建立的任何服務用戶端都會自動發出遙測。若要進一步了解如何使用 `Aws.config` 設定全域設定，請參閱 `Aws.config`。

設定自訂遙測供應商

如果您不想使用 OpenTelemetry 做為遙測提供者，適用於 Ruby 的 AWS SDK 會支援您實作自訂提供者。例如，使用適用於 Ruby GitHub 的 AWS SDK 儲存庫中提供的`OTelProvider實作`可能會有所幫助。如需其他內容，請參閱 適用於 Ruby 的 AWS SDK API 參考`Module: Aws::Telemetry`中的 備註。

跨度屬性

追蹤是遙測的輸出。追蹤包含一或多個跨度。範圍具有包含其他中繼資料的屬性，這些中繼資料會在適合方法呼叫時自動包含。以下是適用於 Ruby 的 SDK 支援的屬性清單，其中：

- 屬性名稱 - 用於標記顯示在追蹤中資料的名稱。
- 類型 - 值的資料類型。
- 描述 - 值所代表內容的描述。

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

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

> **Tip**
>
> OpenTelemetry-Ruby 有其他實作與適用於 Ruby 現有遙測支援的 SDK 整合。如需詳細資訊，請參閱 `open-telemetry` GitHub 儲存庫中的 OpenTelemetry AWS-SDK Instrumentation。

在適用於 Ruby 的 AWS SDK 中設定 HTTP 層級設定

設定非標準端點

區域用於建構 SSL 端點以用於 AWS 請求。如果您需要在所選區域中使用非標準端點，請將`endpoint`項目新增至 `Aws.config`。或者，在建立服務用戶端或資源物件`endpoint:`時設定 。下列範例會在`other_endpoint`端點中建立 Amazon S3 資源物件。

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

若要針對 API 請求使用您選擇的端點，並讓該選擇持續存在，請參閱 AWS SDKs和工具參考指南中的服務特定端點組態選項。

使用適用於 Ruby 的 AWS 開發套件

本節提供有關使用適用於 Ruby 的 AWS SDK 開發軟體的資訊，包括如何使用某些 SDK 的進階功能。

AWS SDKs和工具參考指南也包含許多 AWS SDKs 中常見的設定、功能和其他基本概念。

主題

- 使用適用於 Ruby 的 AWS SDK 提出 AWS 服務 請求
- 使用適用於 Ruby REPL 的 AWS SDK 公用程式
- 使用適用於 Ruby 的 AWS SDK 搭配 Ruby on Rails
- 使用適用於 Ruby 的 AWS SDK 用戶端的線路追蹤資訊進行偵錯
- 使用 stubbing 將測試新增至適用於 Ruby 的 AWS SDK 應用程式
- 在適用於 Ruby 的 AWS SDK 中使用分頁結果
- 在適用於 Ruby 的 AWS SDK 中使用等待程式

使用適用於 Ruby 的 AWS SDK 提出 AWS 服務 請求

若要以程式設計方式存取 AWS 服務，SDKs會為每個 使用用戶端類別 AWS 服務。例如，如果您的應用程式需要存取 Amazon EC2，您的應用程式會建立 Amazon EC2 用戶端物件以與該服務連接。然後，您可以使用 服務用戶端向該用戶端提出請求 AWS 服務。

若要向 提出請求 AWS 服務，您必須先建立和設定服務用戶端。對於 AWS 服務 每個程式碼使用的 ，它都有自己的 Gem 套件和自己的專用類型來與其互動。用戶端會針對服務公開的每個 API 操作公開一個方法。

每個服務用戶端都需要 AWS 區域 和登入資料提供者。SDK 使用這些值將請求傳送到 資源的正確區域，並使用正確的登入資料簽署請求。您可以在程式碼中以程式設計方式指定這些值，或從環境自動載入這些值。

- 執行個體化用戶端類別時，必須提供 AWS 憑證。如需 SDK 檢查身分驗證提供者的順序，請參閱 登入資料提供者鏈結。

- 開發套件有一系列位置 （或來源） 可供其檢查，以尋找組態設定的值。如需詳細資訊，請參閱設定的優先順序。

適用於 Ruby 的 SDK 包含提供 界面的用戶端類別 AWS 服務。每個用戶端類別都支援特定 AWS 服務 ，並遵循慣例 `Aws::<service identifier>::Client`。例如， `Aws::S3::Client` 提供 Amazon Simple Storage Service 服務的界面，`Aws::SQS::Client`並提供 Amazon Simple Queue Service 服務的界面。

所有 的所有用戶端類別 AWS 服務 都是執行緒安全的。

您可以直接將組態選項傳遞給用戶端和資源建構函式。這些選項優先於環境和`Aws.config`預設值。

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

使用適用於 Ruby REPL 的 AWS SDK 公用程式

`aws-sdk` Gem 套件包含 Read-Eval-Print-Loop (REPL) 互動式命令列界面，您可以在其中測試適用於 Ruby 的 SDK，並立即查看結果。適用於 Ruby Gem 的 SDK 可在 RubyGems.org 取得。

先決條件

- 安裝適用於 Ruby 的 AWS SDK.
- `aws-v3.rb` 位於 `aws-sdk-resources` Gem 套件中。Gem `aws-sdk-resources`套件也包含在主要 `aws-sdk` Gem 套件中。
- 您需要 xml 程式庫，例如 `rexml` Gem。
- 雖然此程式可搭配互動式 Ruby Shell (`irb`) 使用，但我們建議您安裝 `pry` Gem，以提供更強大的 REPL 環境。

Bundler 設定

如果您使用 Bundler，以下更新`Gemfile`將解決先決條件 Gem 套件：

1. 開啟您在安裝適用於 Ruby 的 AWS SDK 時`Gemfile`建立的 。在檔案中新增下列各行：

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

2. 儲存 Gemfile。
3. 安裝 中指定的相依性`Gemfile`：

```
$ bundle install
```

執行 REPL

您可以從aws-v3.rb命令列執行 來存取 REPL。

```
aws-v3.rb
```

或者，您也可以設定詳細旗標來啟用 HTTP 線路記錄。HTTP 線路記錄提供適用於 Ruby 的 AWS SDK 與 之間通訊的相關資訊 AWS。請注意，詳細旗標也會增加額外負荷，讓您的程式碼執行速度變慢。

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

適用於 Ruby 的 SDK 包含用戶端類別，可提供 界面 AWS 服務。每個用戶端類別都支援特定 AWS 服務。在 REPL 中，每個服務類別都有一個協助程式，會傳回新的用戶端物件，以便與該服務互動。協助程式的名稱將是轉換為小寫的服務名稱。例如，Amazon S3 和 Amazon EC2 協助程式物件的名稱ec2分別為 s3和 。若要列出帳戶中的 Amazon S3 儲存貯體，您可以在提示s3.list_buckets中輸入 。

您可以在 REPL 提示quit中輸入 以結束。

使用適用於 Ruby 的 AWS SDK 搭配 Ruby on Rails

Ruby on Rails 提供 Web 開發架構，讓您輕鬆使用 Ruby 建立網站。

AWS 提供 aws-sdk-rails Gem 套件，可輕鬆與 Rails 整合。您可以使用 AWS Elastic Beanstalk AWS OpsWorks AWS CodeDeploy或 AWS Rails Provisioner 在 AWS 雲端中部署和執行 Rails 應用程式。

如需安裝和使用 aws-sdk-rails Gem 套件的資訊，請參閱 GitHub 儲存庫 https：//https://github.com/aws/aws-sdk-rails。

使用適用於 Ruby 的 AWS SDK 用戶端的線路追蹤資訊進行偵錯

您可以設定http_wire_trace布林值，從 AWS 用戶端取得線路追蹤資訊。線路追蹤資訊有助於區分用戶端變更、服務問題和使用者錯誤。當 時true，設定會顯示正在線路上傳送的內容。下列範例會建立 Amazon S3 用戶端，並在建立用戶端時啟用線索追蹤。

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

根據下列程式碼和引數 bucket_name，輸出會顯示訊息，指出具有該名稱的儲存貯體是否存在。

```
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
```

如果儲存貯體存在，則輸出類似於以下內容。（為便於閱讀，已將 傳回新增至 HEAD 行。）

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

您也可以在建立用戶端後開啟線路追蹤。

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

如需回報之線路追蹤資訊中欄位的詳細資訊，請參閱 Transfer Family 必要請求標頭。

使用 stubbing 將測試新增至適用於 Ruby 的 AWS SDK 應用程式

了解如何在適用於 Ruby 的 AWS SDK 應用程式中清除用戶端回應和用戶端錯誤。

繪製用戶端回應

當您清除回應時，適用於 Ruby 的 AWS SDK 會停用網路流量，而用戶端會傳回已清除 （或仿造） 的資料。如果您不提供截斷的資料，用戶端會傳回：

- 列為空白陣列
- 映射為空雜湊
- 數值為零
- 日期為 now

下列範例會傳回 Amazon S3 儲存貯體清單的虛設名稱。

```
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
```

執行此程式碼會顯示下列項目。

```
aws-sdk
```

```
aws-sdk2
```

> **Note**
>
> 在您提供任何截斷的資料之後，預設值不會再套用至任何剩餘的執行個體屬性。這表示在先前的範例中，剩餘的執行個體屬性 不是 `creation_date`，now而是 `nil`。

適用於 Ruby 的 AWS 開發套件會驗證您的虛設資料。如果您傳入錯誤類型的資料，就會引發`ArgumentError`例外狀況。例如，如果 不是先前的 指派`bucket_data`，而是使用下列項目：

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

適用於 Ruby 的 AWS SDK 引發兩個`ArgumentError`例外狀況。

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

繪製用戶端錯誤

您也可以清除適用於 Ruby 的 AWS SDK 針對特定方法引發的錯誤。下列範例顯示 `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
```

在適用於 Ruby 的 AWS SDK 中使用分頁結果

當承載太大而無法在單一回應中傳回時，許多 AWS 操作會傳回截斷的結果。反之，服務會傳回一部分的資料和字符，以擷取下一組項目。此模式稱為分頁。

分頁回應可列舉

處理分頁回應資料最簡單的方法是在回應物件中使用內建列舉器，如下列範例所示。

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

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

這會為每個 API 呼叫產生一個回應物件，並列舉具名儲存貯體中的物件。SDK 會擷取其他頁面的資料以完成請求。

手動處理分頁回應

若要自行處理分頁，請使用回應的 `next_page?`方法來驗證要擷取的頁面更多，或使用 `last_page?`方法來驗證沒有更多頁面要擷取。

如果有更多頁面，請使用 `next_page` (請注意沒有 ?) 方法來擷取結果的下一頁，如下列範例所示。

```
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**
>
> 如果您呼叫 `next_page`方法，但沒有更多頁面可供擷取，開發套件會引發 Aws：：PageableResponse：：LastPageError 例外狀況。

分頁資料類別

適用於 Ruby 的 AWS SDK 中的分頁資料由 Aws：：PageableResponse 類別處理，該類別包含在 Seahorse：：Client：：Response 中，以提供分頁資料的存取權。

在適用於 Ruby 的 AWS SDK 中使用等待程式

等待程式是輪詢用戶端上特定狀態的公用程式方法。在為服務用戶端定義的輪詢間隔嘗試數次之後，等待程式可能會失敗。如需如何使用等待程式的範例，請參閱 AWS 程式碼範例儲存庫中 Amazon DynamoDB 加密用戶端的 create_table 方法。

叫用等待程式

若要叫用等待程式，請在服務用戶端`wait_until`上呼叫 。在下列範例中，等待程式會等待執行個體`i-12345678`執行，然後再繼續。

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

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

第一個參數是等待程式名稱，其專屬於服務用戶端，並指出正在等待哪個操作。第二個參數是傳遞至等待程式呼叫之用戶端方法的參數雜湊，其會根據等待程式名稱而有所不同。

如需可等待的操作清單，以及針對每個操作呼叫的用戶端方法，請參閱您正在使用的用戶端的`waiter_names`和 `wait_until` 欄位文件。

等待失敗

等待程式可能會失敗，但有下列任何例外狀況。

Aws：：Waiters：：Errors：：FailureStateError

等待時遇到失敗狀態。

Aws：：Waiters：：Errors：：NoSuchWaiterError

指定的等待程式名稱不會為正在使用的用戶端定義。

Aws：：Waiters：：Errors：：TooManyAttemptsError

嘗試次數超過等待程式`max_attempts`的值。

Aws：：Waiters：：Errors：：UnexpectedError

等待時發生非預期的錯誤。

Aws：：Waiters：：Errors：：WaiterFailed

超過其中一個等待狀態，或在等待時發生另一個失敗。

除了 之外，所有這些錯誤NoSuchWaiterError都是以 為基礎WaiterFailed。若要擷取等待程式中的錯誤，請使用 WaiterFailed，如下列範例所示。

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

設定等待程式

每個等待程式都有預設輪詢間隔，以及在將控制權傳回至您的程式之前嘗試的最大次數。若要設定這些值，請在 wait_until呼叫中使用 max_attempts和 delay: 參數。下列範例會等待最多 25 秒，每 5 秒輪詢一次。

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

若要停用等待失敗，請將其中一個參數的值設定為 nil。

擴展等待程式

若要修改等待者的行為，您可以註冊在每次輪詢嘗試之前和等待之前觸發的回呼。

下列範例透過將每次嘗試的等待時間加倍，在等待程式中實作指數退避。

```
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
```

下列範例會停用最大嘗試次數，並改為等待一小時 (3600 秒） 再失敗。

```
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
```

適用於 Ruby 的 SDK 程式碼範例

本主題中的程式碼範例說明如何 適用於 Ruby 的 AWS SDK 搭配 使用 AWS。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

有些服務包含其他範例類別，示範如何利用特定於服務的程式庫或函數。

服務

- 適用於 Ruby 的 SDK 的 Aurora 範例
- 使用適用於 Ruby 的 SDK 的 Auto Scaling 範例
- 使用適用於 Ruby 的 SDK 的 CloudTrail 範例
- 使用適用於 Ruby 的 SDK 的 CloudWatch 範例
- 使用適用於 Ruby 的 SDK 的 Amazon Cognito 身分提供者範例
- 使用適用於 Ruby 的 SDK 的 Amazon Comprehend 範例
- 使用適用於 Ruby 的 SDK 的 Amazon DocumentDB 範例
- 適用於 Ruby 的 SDK 的 DynamoDB 範例
- 使用適用於 Ruby 的 SDK 的 Amazon EC2 範例
- 使用適用於 Ruby 的 SDK 的 Elastic Beanstalk 範例
- 使用適用於 Ruby 的 SDK 的 EventBridge 範例
- AWS Glue 使用適用於 Ruby 的 SDK 的範例
- 使用適用於 Ruby 的 SDK 的 IAM 範例
- 使用適用於 Ruby 的 SDK 的 Kinesis 範例
- AWS KMS 使用適用於 Ruby 的 SDK 的範例
- 使用適用於 Ruby 的 SDK 的 Lambda 範例
- 使用適用於 Ruby 的 SDK 的 Amazon MSK 範例
- 使用適用於 Ruby 的 SDK 的 Amazon Polly 範例
- 使用適用於 Ruby 的 SDK 的 Amazon RDS 範例
- 使用適用於 Ruby 的 SDK 的 Amazon S3 範例

- 使用適用於 Ruby 的 SDK 的 Amazon SES 範例
- 使用適用於 Ruby 的 SDK 的 Amazon SES API v2 範例
- 使用適用於 Ruby 的 SDK 的 Amazon SNS 範例
- 使用適用於 Ruby 的 SDK 的 Amazon SQS 範例
- AWS STS 使用適用於 Ruby 的 SDK 的範例
- 使用適用於 Ruby 的 SDK 的 Amazon Textract 範例
- 使用適用於 Ruby 的 SDK 的 Amazon Translate 範例

適用於 Ruby 的 SDK 的 Aurora 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Aurora 執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用

開始使用

Hello Aurora

下列程式碼範例示範如何開始使用 Aurora。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBClusters。

使用適用於 Ruby 的 SDK 的 Auto Scaling 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Auto Scaling 來執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用

開始使用

Hello Auto Scaling

下列程式碼範例說明如何開始使用 Auto Scaling。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeAutoScalingGroups。

使用適用於 Ruby 的 SDK 的 CloudTrail 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 CloudTrail 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateTrail`

以下程式碼範例顯示如何使用 `CreateTrail`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateTrail。

`DeleteTrail`

以下程式碼範例顯示如何使用 `DeleteTrail`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteTrail。

`ListTrails`

以下程式碼範例顯示如何使用 `ListTrails`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListTrails。

`LookupEvents`

以下程式碼範例顯示如何使用 `LookupEvents`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 LookupEvents。

使用適用於 Ruby 的 SDK 的 CloudWatch 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 CloudWatch 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`DescribeAlarms`

以下程式碼範例顯示如何使用 `DescribeAlarms`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱適用於 Ruby 的 AWS SDK API 參考中的 DescribeAlarms。

`DescribeAlarmsForMetric`

以下程式碼範例顯示如何使用 `DescribeAlarmsForMetric`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
#
# @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

      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__
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 DescribeAlarmsForMetric。

`DisableAlarmActions`

以下程式碼範例顯示如何使用 `DisableAlarmActions`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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__
```

- 如需詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 DisableAlarmActions。

`ListMetrics`

以下程式碼範例顯示如何使用 `ListMetrics`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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__
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 ListMetrics。

`PutMetricAlarm`

以下程式碼範例顯示如何使用 `PutMetricAlarm`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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 theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
```

```
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 PutMetricAlarm。

PutMetricData

以下程式碼範例顯示如何使用 PutMetricData。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 PutMetricData。

使用適用於 Ruby 的 SDK 的 Amazon Cognito 身分提供者範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon Cognito Identity Provider 執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用

開始使用

Hello Amazon Cognito

下列程式碼範例顯示如何開始使用 Amazon Cognito。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 ListUserPools。

使用適用於 Ruby 的 SDK 的 Amazon Comprehend 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon Comprehend 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立應用程式以分析客戶意見回饋

下列程式碼範例會示範如何建立可分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案的應用程式。

SDK for Ruby

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。

- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署的說明，請參閱 GitHub 中的專案。

此範例中使用的服務

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

使用適用於 Ruby 的 SDK 的 Amazon DocumentDB 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon DocumentDB 執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 無伺服器範例

無伺服器範例

使用 Amazon DocumentDB 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 DocumentDB 變更串流的記錄來接收所觸發的事件。函數會擷取 DocumentDB 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 使用 Amazon DocumentDB 事件。

```
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
```

適用於 Ruby 的 SDK 的 DynamoDB 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 DynamoDB 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念

- 動作
- 案例
- 無伺服器範例

開始使用

Hello DynamoDB

下列程式碼範例示範如何開始使用 DynamoDB。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListTables。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立可存放電影資料的資料表。
- 放入、取得和更新資料表中的單個電影。
- 將影片資料從範例 JSON 檔案寫入資料表。
- 查詢特定年份發表的電影。
- 掃描某個年份範圍內發表的電影。
- 從資料表刪除電影，然後刪除資料表。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立封裝 DynamoDB 資料表的類別。

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

建立 Helper 函數以下載並擷取範例 JSON 檔案。

```
# 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
```

執行互動式案例以建立資料表並對其執行動作。

```
  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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

動作

`BatchExecuteStatement`

以下程式碼範例顯示如何使用 `BatchExecuteStatement`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

使用 PartiQL 讀取一批項目。

```
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
```

使用 PartiQL 刪除一批項目。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 BatchExecuteStatement。

`BatchWriteItem`

以下程式碼範例顯示如何使用 `BatchWriteItem`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 BatchWriteItem。

`CreateTable`

以下程式碼範例顯示如何使用 `CreateTable`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateTable。

DeleteItem

以下程式碼範例顯示如何使用 DeleteItem。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteItem。

`DeleteTable`

以下程式碼範例顯示如何使用 `DeleteTable`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteTable。

`DescribeTable`

以下程式碼範例顯示如何使用 `DescribeTable`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeTable。

`ExecuteStatement`

以下程式碼範例顯示如何使用 `ExecuteStatement`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

使用 PartiQL 選取單一項目。

```
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
```

使用 PartiQL 更新單一項目。

```
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
```

使用 PartiQL 新增單一項目。

```
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 刪除單一項目。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ExecuteStatement。

`GetItem`

以下程式碼範例顯示如何使用 `GetItem`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetItem。

`ListTables`

以下程式碼範例顯示如何使用 `ListTables`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

判斷資料表是否存在。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 ListTables。

`PutItem`

以下程式碼範例顯示如何使用 `PutItem`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 PutItem。

Query

以下程式碼範例顯示如何使用 Query。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 Query。

Scan

以下程式碼範例顯示如何使用 Scan。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

```
      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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 Scan。

`UpdateItem`

以下程式碼範例顯示如何使用 `UpdateItem`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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] },
```

```
      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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 UpdateItem。

案例

使用多批 PartiQL 陳述式查詢資料表

以下程式碼範例顯示做法：

- 透過執行多個 SELECT 陳述式取得一批項目。
- 透過執行多個 INSERT 陳述式新增一批項目。
- 透過執行多個 UPDATE 陳述式更新一批項目。
- 透過執行多個 DELETE 陳述式刪除一批項目。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

執行一個情境，該情境會建立資料表並執行批次 PartiQL 查詢。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 BatchExecuteStatement。

使用 PartiQL 查詢資料表

以下程式碼範例顯示做法：

- 透過執行 SELECT 陳述式取得項目。

- 透過執行 INSERT 陳述式新增項目。
- 透過執行 UPDATE 陳述式更新項目。
- 透過執行 DELETE 陳述式刪除項目。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

執行一個情境，該情境建立資料表並執行 PartiQL 查詢。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ExecuteStatement。

無伺服器範例

使用 DynamoDB 觸發條件調用 Lambda 函式

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 DynamoDB 串流的記錄來接收所觸發的事件。函數會擷取 DynamoDB 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 DynamoDB 事件。

```
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
```

使用 DynamoDB 觸發條件報告 Lambda 函式的批次項目失敗

下列程式碼範例示範如何針對接收來自 DynamoDB 串流之事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 報告 DynamoDB 批次項目失敗。

```
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
```

使用適用於 Ruby 的 SDK 的 Amazon EC2 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon EC2 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 動作

開始使用

您好 Amazon EC2

下列程式碼範例示範如何開始使用 Amazon EC2。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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.
  #
```

```
  # @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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeSecurityGroups。

動作

`AllocateAddress`

以下程式碼範例顯示如何使用 `AllocateAddress`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 AllocateAddress。

`AssociateAddress`

以下程式碼範例顯示如何使用 `AssociateAddress`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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 assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
```

```
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 AssociateAddress。

`CreateKeyPair`

以下程式碼範例顯示如何使用 `CreateKeyPair`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
rescue StandardError => e
```

```
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateKeyPair。

`CreateRouteTable`

以下程式碼範例顯示如何使用 `CreateRouteTable`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 CreateRouteTable。

`CreateSecurityGroup`

以下程式碼範例顯示如何使用 `CreateSecurityGroup`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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..."
  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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateSecurityGroup。

`CreateSubnet`

以下程式碼範例顯示如何使用 `CreateSubnet`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 CreateSubnet。

`CreateVpc`

以下程式碼範例顯示如何使用 `CreateVpc`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 CreateVpc。

`DescribeInstances`

以下程式碼範例顯示如何使用 `DescribeInstances`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeInstances。

`DescribeRegions`

以下程式碼範例顯示如何使用 `DescribeRegions`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeRegions。

`ReleaseAddress`

以下程式碼範例顯示如何使用 `ReleaseAddress`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ReleaseAddress。

`StartInstances`

以下程式碼範例顯示如何使用 `StartInstances`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 StartInstances。

`StopInstances`

以下程式碼範例顯示如何使用 `StopInstances`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

  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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 StopInstances。

`TerminateInstances`

以下程式碼範例顯示如何使用 `TerminateInstances`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 TerminateInstances。

使用適用於 Ruby 的 SDK 的 Elastic Beanstalk 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Elastic Beanstalk 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`DescribeApplications`

以下程式碼範例顯示如何使用 `DescribeApplications`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeApplications。

`ListAvailableSolutionStacks`

以下程式碼範例顯示如何使用 `ListAvailableSolutionStacks`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

```
  # 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListAvailableSolutionStacks。

`UpdateApplication`

以下程式碼範例顯示如何使用 `UpdateApplication`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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)
```

```
    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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 UpdateApplication。

使用適用於 Ruby 的 SDK 的 EventBridge 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 EventBridge 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立和觸發規則

下列程式碼範例示範如何在 Amazon EventBridge 中建立和觸發規則。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

以正確的順序呼叫函數。

```
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'
```

檢查提供給此函數的主題中是否存在指定的 Amazon Simple Notification Service (Amazon SNS) 主題。

```
# 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
```

檢查 Amazon SNS 中呼叫者可用的主題中是否存在指定的主題。

```
# 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 中建立主題，然後訂閱電子郵件地址以接收該主題的通知。

```
# 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
```

檢查提供給此函數的 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
```

檢查 IAM 中呼叫者可用的角色中是否存在指定的角色。

```
# 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 中建立角色。

```
# 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
```

檢查提供給此函數的規則中是否存在指定的 EventBridge 規則。

```
# 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
```

檢查 EventBridge 中呼叫者可用的規則中是否存在指定的規則。

```
# 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 EventBridge 中建立規則。

```
# 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
```

檢查 Amazon CloudWatch Logs 中可供呼叫者使用的日誌群組中是否存在指定的日誌群組。

```
# 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 CloudWatch Logs 中建立日誌群組。

```
# 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 CloudWatch Logs 中將事件寫入日誌串流。

```
# 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
```

重新啟動 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體並將相關活動的資訊新增至 CloudWatch Logs 中的日誌串流。

```
# 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
```

顯示 EventBridge 中規則的活動相關資訊。

```
# 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
```

顯示 CloudWatch Logs 日誌群組中所有日誌串流的日誌資訊。

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
```

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

向發起人顯示提醒，以手動清除他們不再需要的任何相關 AWS 資源。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - PutEvents
  - PutRule

AWS Glue 使用適用於 Ruby 的 SDK 的範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 來執行動作和實作常見案例 AWS Glue。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

您好 AWS Glue

下列程式碼範例示範如何開始使用 AWS Glue。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
end

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListJobs。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立網路爬取公有 Amazon S3 儲存貯體的爬蟲程式，以及產生 CSV 格式中繼資料的資料庫。
- 列出 中資料庫和資料表的相關資訊 AWS Glue Data Catalog。

- 建立從 S3 儲存貯體中擷取 CSV 資料的任務、轉換資料，以及將 JSON 格式的輸出載入至另一個 S3 儲存貯體。
- 列出任務執行的相關資訊、檢視已轉換的資料以及清除資源。

如需詳細資訊，請參閱教學課程： AWS Glue Studio 入門。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立可包裝案例中所用 AWS Glue 函數的類別。

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

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

```
    raise
  end

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

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

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

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

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

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

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

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

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

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

```
  end

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

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

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

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

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

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

建立可執行案例的類別。

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

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
```

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

  private

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

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

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

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
```

```
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

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

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

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

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

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

  puts 'Demo complete.'
end
```

建立 ETL 指令碼，AWS Glue 供 在任務執行期間擷取、轉換和載入資料。

```
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()
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases

- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

動作

`CreateCrawler`

以下程式碼範例顯示如何使用 `CreateCrawler`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateCrawler。

`CreateJob`

以下程式碼範例顯示如何使用 `CreateJob`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateJob。

DeleteCrawler

以下程式碼範例顯示如何使用 DeleteCrawler。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteCrawler。

`DeleteDatabase`

以下程式碼範例顯示如何使用 `DeleteDatabase`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteDatabase。

`DeleteJob`

以下程式碼範例顯示如何使用 `DeleteJob`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteJob。

`DeleteTable`

以下程式碼範例顯示如何使用 `DeleteTable`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteTable。

`GetCrawler`

以下程式碼範例顯示如何使用 `GetCrawler`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetCrawler。

GetDatabase

以下程式碼範例顯示如何使用 GetDatabase。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetDatabase。

GetJobRun

以下程式碼範例顯示如何使用 GetJobRun。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetJobRun。

GetJobRuns

以下程式碼範例顯示如何使用 GetJobRuns。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetJobRuns。

`GetTables`

以下程式碼範例顯示如何使用 `GetTables`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetTables。

`ListJobs`

以下程式碼範例顯示如何使用 `ListJobs`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListJobs。

`StartCrawler`

以下程式碼範例顯示如何使用 `StartCrawler`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 StartCrawler。

`StartJobRun`

以下程式碼範例顯示如何使用 `StartJobRun`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 StartJobRun。

使用適用於 Ruby 的 SDK 的 IAM 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 IAM 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello IAM

下列程式碼範例說明如何開始使用 IAM。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListPolicies。

基本概念

了解基本概念

下列程式碼範例示範如何建立使用者並擔任角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立沒有許可的使用者。
- 建立一個可授予許可的角色，以列出帳戶的 Amazon S3 儲存貯體。
- 新增政策，讓使用者擔任該角色。
- 使用暫時憑證，擔任角色並列出 S3 儲存貯體，然後清理資源。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

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

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

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

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

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

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
```

```
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
```

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

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated into
a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
```

```
 # Because the resource uses credentials with limited access, it may not be able to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
     break if count.zero?
   end
 rescue Aws::Errors::ServiceError => e
   if e.code == 'AccessDenied'
     puts('Attempt to list buckets with no permissions: AccessDenied.')
   else
     @logger.info("Couldn't list buckets for the account. Here's why: ")
     @logger.info("\t#{e.code}: #{e.message}")
     raise
   end
 end

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
```

```
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
    end

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

```
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

動作

`AttachRolePolicy`

以下程式碼範例顯示如何使用 `AttachRolePolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、連接和分離角色政策。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
```

```
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
```

```
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 AttachRolePolicy。

`AttachUserPolicy`

以下程式碼範例顯示如何使用 `AttachUserPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 AttachUserPolicy。

`CreateAccessKey`

以下程式碼範例顯示如何使用 `CreateAccessKey`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、停用和刪除存取金鑰。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreateAccessKey。

`CreateAccountAlias`

以下程式碼範例顯示如何使用 `CreateAccountAlias`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、建立和刪除帳戶別名。

```
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}")
    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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreateAccountAlias。

`CreatePolicy`

以下程式碼範例顯示如何使用 `CreatePolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、連接和分離角色政策。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
 #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
 #{e.message}")
    raise
```

```
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
```

```
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreatePolicy。

`CreateRole`

以下程式碼範例顯示如何使用 `CreateRole`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
 # 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
```

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

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreateRole。

`CreateServiceLinkedRole`

以下程式碼範例顯示如何使用 `CreateServiceLinkedRole`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreateServiceLinkedRole。

`CreateUser`

以下程式碼範例顯示如何使用 `CreateUser`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 CreateUser。

`DeleteAccessKey`

以下程式碼範例顯示如何使用 `DeleteAccessKey`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、停用和刪除存取金鑰。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteAccessKey。

`DeleteAccountAlias`

以下程式碼範例顯示如何使用 `DeleteAccountAlias`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、建立和刪除帳戶別名。

```
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}")
    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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteAccountAlias。

DeleteRole

以下程式碼範例顯示如何使用 DeleteRole。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteRole。

DeleteServerCertificate

以下程式碼範例顯示如何使用 DeleteServerCertificate。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、更新和刪除伺服器憑證。

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
```

```
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteServerCertificate。

DeleteServiceLinkedRole

以下程式碼範例顯示如何使用 DeleteServiceLinkedRole。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteServiceLinkedRole。

`DeleteUser`

以下程式碼範例顯示如何使用 `DeleteUser`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteUser。

`DeleteUserPolicy`

以下程式碼範例顯示如何使用 `DeleteUserPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DeleteUserPolicy。

`DetachRolePolicy`

以下程式碼範例顯示如何使用 `DetachRolePolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、連接和分離角色政策。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
```

```
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

```
  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DetachRolePolicy。

`DetachUserPolicy`

以下程式碼範例顯示如何使用 `DetachUserPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 DetachUserPolicy。

`GetAccountPasswordPolicy`

以下程式碼範例顯示如何使用 `GetAccountPasswordPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 GetAccountPasswordPolicy。

`GetPolicy`

以下程式碼範例顯示如何使用 `GetPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 GetPolicy。

GetRole

以下程式碼範例顯示如何使用 GetRole。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 GetRole。

GetUser

以下程式碼範例顯示如何使用 GetUser。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
 # 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetUser。

`ListAccessKeys`

以下程式碼範例顯示如何使用 `ListAccessKeys`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、停用和刪除存取金鑰。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListAccessKeys。

ListAccountAliases

以下程式碼範例顯示如何使用 ListAccountAliases。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、建立和刪除帳戶別名。

```
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}")
```

```
    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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListAccountAliases。

`ListAttachedRolePolicies`

以下程式碼範例顯示如何使用 `ListAttachedRolePolicies`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、連接和分離角色政策。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
```

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

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListAttachedRolePolicies。

`ListGroups`

以下程式碼範例顯示如何使用 `ListGroups`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListGroups。

`ListPolicies`

以下程式碼範例顯示如何使用 `ListPolicies`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

此範例模組會列出、建立、連接和分離角色政策。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
```

```
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

```
  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListPolicies。

`ListRolePolicies`

以下程式碼範例顯示如何使用 `ListRolePolicies`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
  # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListRolePolicies。

`ListRoles`

以下程式碼範例顯示如何使用 `ListRoles`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListRoles。

`ListSAMLProviders`

以下程式碼範例顯示如何使用 `ListSAMLProviders`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListSAMLProviders。

`ListServerCertificates`

以下程式碼範例顯示如何使用 `ListServerCertificates`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、更新和刪除伺服器憑證。

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
```

```
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListServerCertificates。

`ListUsers`

以下程式碼範例顯示如何使用 `ListUsers`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 ListUsers。

`PutUserPolicy`

以下程式碼範例顯示如何使用 `PutUserPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 PutUserPolicy。

`UpdateServerCertificate`

以下程式碼範例顯示如何使用 `UpdateServerCertificate`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

列出、更新和刪除伺服器憑證。

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end
```

```
    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 UpdateServerCertificate。

UpdateUser

以下程式碼範例顯示如何使用 UpdateUser。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 UpdateUser。

使用適用於 Ruby 的 SDK 的 Kinesis 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Kinesis 執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 無伺服器範例

無伺服器範例

使用 Kinesis 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 Kinesis 串流的記錄來接收所觸發的事件。此函數會擷取 Kinesis 承載、從 Base64 解碼，並記錄記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 Kinesis 事件。

```
# 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
```

使用 Kinesis 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何針對接收來自 Kinesis 串流之事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來報告 Kinesis 批次項目失敗。

```
# 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 使用適用於 Ruby 的 SDK 的範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 來執行動作和實作常見案例 AWS KMS。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateKey`

以下程式碼範例顯示如何使用 `CreateKey`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateKey。

`Decrypt`

以下程式碼範例顯示如何使用 `Decrypt`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 Decrypt。

`Encrypt`

以下程式碼範例顯示如何使用 `Encrypt`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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*')
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 Encrypt。

`ReEncrypt`

以下程式碼範例顯示如何使用 `ReEncrypt`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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*')
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ReEncrypt。

使用適用於 Ruby 的 SDK 的 Lambda 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Lambda 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作
- 案例
- 無伺服器範例

開始使用

Hello Lambda

下列程式碼範例示範如何開始使用 Lambda。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListFunctions。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 IAM 角色和 Lambda 函數，然後上傳處理常式程式碼。
- 調用具有單一參數的函數並取得結果。
- 更新函數程式碼並使用環境變數進行設定。
- 調用具有新參數的函數並取得結果。顯示傳回的執行日誌。

- 列出您帳戶的函數，然後清理相關資源。

如需詳細資訊，請參閱使用主控台建立 Lambda 函數。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

為能夠寫入日誌的 Lambda 函數設定先決條件 IAM 許可。

```
# 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
```

定義 Lambda 處理常式，該處理常式以作為調用參數提供的數字遞增。

```
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
```

將您的 Lambda 函數壓縮到部署套件中。

```
  # 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
```

建立新 Lambda 函數。

```
  # 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
```

使用選用的執行階段參數調用 Lambda 函數。

```
  # 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 函數的組態以注入新的環境變數。

```
  # 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
```

使用包含不同程式碼的不同部署套件來更新 Lambda 函數的程式碼。

```
  # 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
```

使用內建分頁程式列出所有現有的 Lambda 函數。

```
  # 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
```

刪除特定 Lambda 函數。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

動作

`CreateFunction`

以下程式碼範例顯示如何使用 `CreateFunction`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的「CreateFunction」。

`DeleteFunction`

以下程式碼範例顯示如何使用 `DeleteFunction`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

  # 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteFunction。

`GetFunction`

以下程式碼範例顯示如何使用 `GetFunction`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetFunction。

`Invoke`

以下程式碼範例顯示如何使用 `Invoke`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 Invoke。

`ListFunctions`

以下程式碼範例顯示如何使用 `ListFunctions`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListFunctions。

`UpdateFunctionCode`

以下程式碼範例顯示如何使用 `UpdateFunctionCode`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 UpdateFunctionCode。

`UpdateFunctionConfiguration`

以下程式碼範例顯示如何使用 `UpdateFunctionConfiguration`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 UpdateFunctionConfiguration。

案例

建立應用程式以分析客戶意見回饋

下列程式碼範例會示範如何建立可分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案的應用程式。

SDK for Ruby

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。

- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署的說明，請參閱 GitHub 中的專案。

此範例中使用的服務

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

無伺服器範例

連線至 Lambda 函數中的 Amazon RDS 資料庫

以下程式碼範例示範如何實作連線至 RDS 資料庫的 Lambda 函式。該函數會提出簡單的資料庫請求並傳回結果。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 連線至 Lambda 函數中的 Amazon RDS 資料庫。

```
# 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
```

使用 Kinesis 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 Kinesis 串流的記錄來接收所觸發的事件。此函數會擷取 Kinesis 承載、從 Base64 解碼，並記錄記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 Kinesis 事件。

```
# 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
```

使用 DynamoDB 觸發條件調用 Lambda 函式

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 DynamoDB 串流的記錄來接收所觸發的事件。函數會擷取 DynamoDB 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 DynamoDB 事件。

```
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
```

使用 Amazon DocumentDB 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 DocumentDB 變更串流的記錄來接收所觸發的事件。函數會擷取 DocumentDB 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 使用 Amazon DocumentDB 事件。

```
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 Amazon MSK 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 Amazon MSK 叢集的記錄來接收所觸發的事件。函數會擷取 MSK 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來取用 Amazon MSK 事件。

```
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 Amazon S3 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式接收透過上傳物件至 S3 儲存貯體時所觸發的事件。此函數會從事件參數擷取 S3 儲存貯體名稱和物件金鑰，並呼叫 Amazon S3 API 以擷取和記錄物件的內容類型。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 S3 事件。

```
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 Amazon SNS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 SNS 主題的訊息來接收所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 SNS 事件。

```
# 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
```

使用 Amazon SQS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 SQS 佇列的訊息來接收所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 SQS 事件。

```
# 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
```

使用 Kinesis 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何針對接收來自 Kinesis 串流之事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來報告 Kinesis 批次項目失敗。

```
# 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
```

使用 DynamoDB 觸發條件報告 Lambda 函式的批次項目失敗

下列程式碼範例示範如何針對接收來自 DynamoDB 串流之事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 報告 DynamoDB 批次項目失敗。

```
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
```

使用 Amazon SQS 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為接收從 SQS 佇列接收事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 報告 SQS 批次項目失敗。

```
# 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
```

使用適用於 Ruby 的 SDK 的 Amazon MSK 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon MSK 執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 無伺服器範例

無伺服器範例

使用 Amazon MSK 觸發條件調用 Lambda 函數

以下程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 Amazon MSK 叢集的記錄來接收所觸發的事件。函數會擷取 MSK 承載並記下記錄內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來取用 Amazon MSK 事件。

```
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
```

使用適用於 Ruby 的 SDK 的 Amazon Polly 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon Polly 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 案例

動作

`DescribeVoices`

以下程式碼範例顯示如何使用 `DescribeVoices`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeVoices。

`ListLexicons`

以下程式碼範例顯示如何使用 `ListLexicons`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListLexicons。

`SynthesizeSpeech`

以下程式碼範例顯示如何使用 `SynthesizeSpeech`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 SynthesizeSpeech。

案例

建立應用程式以分析客戶意見回饋

下列程式碼範例會示範如何建立可分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案的應用程式。

SDK for Ruby

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署的說明，請參閱 GitHub 中的專案。

此範例中使用的服務

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

使用適用於 Ruby 的 SDK 的 Amazon RDS 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon RDS 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 動作
- 無伺服器範例

開始使用

您好 Amazon RDS

下列程式碼範例說明如何開始使用 Amazon RDS。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBInstances。

動作

`CreateDBSnapshot`

以下程式碼範例顯示如何使用 `CreateDBSnapshot`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateDBSnapshot。

DescribeDBInstances

以下程式碼範例顯示如何使用 DescribeDBInstances。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBInstances。

DescribeDBParameterGroups

以下程式碼範例顯示如何使用 DescribeDBParameterGroups。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBParameterGroups。

`DescribeDBParameters`

以下程式碼範例顯示如何使用 `DescribeDBParameters`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBParameters。

`DescribeDBSnapshots`

以下程式碼範例顯示如何使用 `DescribeDBSnapshots`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DescribeDBSnapshots。

無伺服器範例

連線至 Lambda 函數中的 Amazon RDS 資料庫

以下程式碼範例示範如何實作連線至 RDS 資料庫的 Lambda 函式。該函數會提出簡單的資料庫請求並傳回結果。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 連線至 Lambda 函數中的 Amazon RDS 資料庫。

```
# 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
```

使用適用於 Ruby 的 SDK 的 Amazon S3 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon S3 執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作
- 案例

- 無伺服器範例

開始使用

您好 Amazon S3

下列程式碼範例說明如何開始使用 Amazon S3。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListBuckets。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立儲存貯體並上傳檔案到該儲存貯體。
- 從儲存貯體下載物件。
- 將物件複製至儲存貯體中的子文件夾。
- 列出儲存貯體中的物件。
- 刪除儲存貯體物件和該儲存貯體。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
```

```
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

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

```
  else
    dest_object
  end

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

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

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

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
```

```
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

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

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

動作

`CopyObject`

以下程式碼範例顯示如何使用 `CopyObject`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

複製物件

```
require 'aws-sdk-s3'

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

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object
```

```
  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

複製物件，然後將伺服器端加密新增至目標物件。

```
require 'aws-sdk-s3'

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

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
```

```
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CopyObject。

`CreateBucket`

以下程式碼範例顯示如何使用 `CreateBucket`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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!'
    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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateBucket。

`DeleteBucket`

以下程式碼範例顯示如何使用 `DeleteBucket`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteBucket。

`DeleteBucketCors`

以下程式碼範例顯示如何使用 `DeleteBucketCors`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteBucketCors。

`DeleteBucketPolicy`

以下程式碼範例顯示如何使用 `DeleteBucketPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteBucketPolicy。

`DeleteObjects`

以下程式碼範例顯示如何使用 `DeleteObjects`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteObjects。

`GetBucketCors`

以下程式碼範例顯示如何使用 `GetBucketCors`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetBucketCors。

`GetBucketPolicy`

以下程式碼範例顯示如何使用 `GetBucketPolicy`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetBucketPolicy。

`GetObject`

以下程式碼範例顯示如何使用 `GetObject`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

取得物件。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

取得物件並報告其伺服器端的加密狀態。

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

```
# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetObject。

`HeadObject`

以下程式碼範例顯示如何使用 `HeadObject`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 HeadObject。

`ListBuckets`

以下程式碼範例顯示如何使用 `ListBuckets`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
end

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

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListBuckets。

`ListObjectsV2`

以下程式碼範例顯示如何使用 `ListObjectsV2`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListObjectsV2。

PutBucketCors

以下程式碼範例顯示如何使用 PutBucketCors。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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}"
    false
  end

end
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 PutBucketCors。

PutBucketPolicy

以下程式碼範例顯示如何使用 PutBucketPolicy。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 PutBucketPolicy。

PutBucketWebsite

以下程式碼範例顯示如何使用 PutBucketWebsite。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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 }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 PutBucketWebsite。

`PutObject`

以下程式碼範例顯示如何使用 `PutObject`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

使用受管的上傳工具上傳檔案 (Object.upload_file)。

```
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.上傳檔案。

```
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.Pet 上傳檔案並新增伺服器端加密。

```
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
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 PutObject。

案例

建立預先簽章 URL

下列程式碼範例示範如何建立適用於 Amazon S3 預先簽署的 URL，並上傳物件。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

run_demo if $PROGRAM_NAME == __FILE__
```

無伺服器範例

使用 Amazon S3 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式接收透過上傳物件至 S3 儲存貯體時所觸發的事件。此函數會從事件參數擷取 S3 儲存貯體名稱和物件金鑰，並呼叫 Amazon S3 API 以擷取和記錄物件的內容類型。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 S3 事件。

```
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
```

使用適用於 Ruby 的 SDK 的 Amazon SES 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon SES 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`GetIdentityVerificationAttributes`

以下程式碼範例顯示如何使用 `GetIdentityVerificationAttributes`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 GetIdentityVerificationAttributes。

`ListIdentities`

以下程式碼範例顯示如何使用 `ListIdentities`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListIdentities。

`SendEmail`

以下程式碼範例顯示如何使用 `SendEmail`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
```

```
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

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

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 SendEmail。

`VerifyEmailIdentity`

以下程式碼範例顯示如何使用 `VerifyEmailIdentity`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 VerifyEmailIdentity。

使用適用於 Ruby 的 SDK 的 Amazon SES API v2 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon SES API v2 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`SendEmail`

以下程式碼範例顯示如何使用 `SendEmail`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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)
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 SendEmail。

使用適用於 Ruby 的 SDK 的 Amazon SNS 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon SNS 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 無伺服器範例

動作

`CreateTopic`

以下程式碼範例顯示如何使用 `CreateTopic`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateTopic。

`ListSubscriptions`

以下程式碼範例顯示如何使用 `ListSubscriptions`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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}"
    exit 1
  end
end
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 ListSubscriptions。

ListTopics

以下程式碼範例顯示如何使用 ListTopics。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 ListTopics。

`Publish`

以下程式碼範例顯示如何使用 `Publish`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的發佈。

`SetTopicAttributes`

以下程式碼範例顯示如何使用 `SetTopicAttributes`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 SetTopicAttributes。

`Subscribe`

以下程式碼範例顯示如何使用 `Subscribe`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

透過電子郵件地址訂閱某個主題。

```
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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》。
- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的訂閱。

無伺服器範例

使用 Amazon SNS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 SNS 主題的訊息來接收所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 SNS 事件。

```
# 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
```

使用適用於 Ruby 的 SDK 的 Amazon SQS 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon SQS 執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 無伺服器範例

動作

`ChangeMessageVisibility`

以下程式碼範例顯示如何使用 `ChangeMessageVisibility`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ChangeMessageVisibility。

`CreateQueue`

以下程式碼範例顯示如何使用 `CreateQueue`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
# 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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 CreateQueue。

`DeleteQueue`

以下程式碼範例顯示如何使用 `DeleteQueue`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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)
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 DeleteQueue。

`ListQueues`

以下程式碼範例顯示如何使用 `ListQueues`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ListQueues。

ReceiveMessage

以下程式碼範例顯示如何使用 ReceiveMessage。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 ReceiveMessage。

SendMessage

以下程式碼範例顯示如何使用 SendMessage。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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__
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 SendMessage。

SendMessageBatch

以下程式碼範例顯示如何使用 SendMessageBatch。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
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
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
```

- 如需 API 詳細資訊，請參閱《適用於 Ruby 的 AWS SDK API 參考》中的 SendMessageBatch。

無伺服器範例

使用 Amazon SQS 觸發條件調用 Lambda 函數

下列程式碼範例示範如何實作 Lambda 函式，該函式會透過接收 SQS 佇列的訊息來接收所觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 SQS 事件。

```
# 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
```

使用 Amazon SQS 觸發條件報告 Lambda 函數的批次項目失敗

下列程式碼範例示範如何為接收從 SQS 佇列接收事件的 Lambda 函式，實作部分批次回應。此函數會在回應中報告批次項目失敗，指示 Lambda 稍後重試這些訊息。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 報告 SQS 批次項目失敗。

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

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}
```

```
    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS 使用適用於 Ruby 的 SDK 的範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 來執行動作和實作常見案例 AWS STS。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`AssumeRole`

以下程式碼範例顯示如何使用 `AssumeRole`。

SDK for Ruby

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
 # 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
```

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API Reference 中的 AssumeRole。

使用適用於 Ruby 的 SDK 的 Amazon Textract 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon Textract 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立應用程式以分析客戶意見回饋

下列程式碼範例會示範如何建立可分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案的應用程式。

SDK for Ruby

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署的說明，請參閱 GitHub 中的專案。

此範例中使用的服務

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

使用適用於 Ruby 的 SDK 的 Amazon Translate 範例

下列程式碼範例示範如何使用 適用於 Ruby 的 AWS SDK 搭配 Amazon Translate 執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立應用程式以分析客戶意見回饋

下列程式碼範例會示範如何建立可分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案的應用程式。

SDK for Ruby

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署的說明，請參閱 GitHub 中的專案。

此範例中使用的服務

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

從適用於 Ruby 的 AWS SDK 第 1 版或第 2 版遷移至適用於 Ruby 的 AWS SDK 第 3 版

本主題包含詳細資訊，可協助您從適用於 Ruby 的 AWS SDK 第 1 版或第 2 版遷移到第 3 版。

Side-by-side用量

您不需要將適用於 Ruby 的 AWS SDK 第 1 版或第 2 版取代為第 3 版。您可以在相同的應用程式中一起使用它們。如需詳細資訊，請參閱此部落格文章。

快速範例如下。

```
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
```

您不需要重寫現有的工作版本 1 或 2 程式碼，即可開始使用版本 3 開發套件。有效的遷移策略是僅針對第 3 版 SDK 撰寫新的程式碼。

一般差異

第 3 版與第 2 版的一個重要方式不同。

- 每個服務都以個別 Gem 套件的形式提供。

第 2 版與第 1 版有幾個重要的不同。

- 不同的根命名空間 –`Aws` 相較於 `AWS`。這會啟用side-by-side用量。
- `Aws.config`– 現在是香草 Ruby 雜湊，而不是方法。
- 嚴格建構函數選項 - 在版本 1 開發套件中建構用戶端或資源物件時，會忽略未知建構函數選項。在第 2 版中，未知的建構函數選項會觸發 `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
```

用戶端差異

第 2 版和第 3 版的用戶端類別之間沒有差異。

在版本 1 和版本 2 之間，用戶端類別的外部差異最小。許多服務用戶端在建置用戶端之後都會有相容的介面。一些重要的差異：

- `Aws::S3::Client` - 第 1 版 Amazon S3 用戶端類別為手動編碼。第 2 版是從服務模型產生。第 2 版中的方法名稱和輸入非常不同。
- `Aws::EC2::Client`- 第 2 版使用輸出清單的複數名稱，第 1 版使用尾碼 `_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`– 第 2 版使用結構化回應，其中第 1 版使用 Vanilla Ruby 雜湊。
- 服務類別重新命名 – 第 2 版針對多個服務使用不同的名稱：
  - `AWS::SimpleWorkflow` 已變成 `Aws::SWF`
  - `AWS::ELB` 已變成 `Aws::ElasticLoadBalancing`
  - `AWS::SimpleEmailService` 已變成 `Aws::SES`
- 用戶端組態選項 – 某些第 1 版組態選項會在第 2 版中重新命名。其他則會移除或取代。以下是主要變更：
  - `:use_ssl` 已移除。第 2 版會在任何地方使用 SSL。若要停用 SSL，您必須設定使用 `:endpoint`的 `http://`。
  - `:ssl_ca_file` 現在是 `:ssl_ca_bundle`

- :ssl_ca_path 現在是 :ssl_ca_directory
- 新增了 :ssl_ca_store。
- :endpoint 現在必須是完全合格的 HTTP 或 HTTPS URI，而不是主機名稱。
- 已移除每個服務:*_port的選項，現在已由 取代:endpoint。
- :user_agent_prefix 現在是 :user_agent_suffix

資源差異

第 2 版和第 3 版中的資源介面之間沒有差異。

第 1 版與第 2 版中的資源介面之間存在顯著差異。第 1 版是完全手動編碼的，其中第 2 版的資源界面是從模型產生。第 2 版的資源界面明顯更一致。一些系統差異包括：

- 個別的資源類別 – 在第 2 版中，服務名稱是模組，而不是類別。在此模組中，它是資源界面：

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- 參考資源 – 第 2 版 SDK 將集合和個別資源取得器分成兩種不同的方法：

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- 批次操作 – 在第 1 版中，所有批次操作都是手動編碼的公用程式。在第 2 版中，許多批次操作都是透過 API 自動產生的批次操作。第 2 版批次處理介面與第 1 版非常不同。

適用於 Ruby 的 AWS SDK 安全性

雲端安全是 Amazon Web Services (AWS) 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。安全性是 AWS 與您之間的共同責任。共同責任模型 將此描述為雲端本身的安全和雲端內部的安全。

Cloud 的安全 – AWS 負責保護執行 AWS 雲端中提供的所有服務的基礎設施，並提供您可以安全使用的服務。我們的安全責任是 的最高優先順序 AWS，我們的安全有效性由第三方稽核人員定期測試和驗證，作為AWS 合規計劃的一部分。

雲端的安全性 – 您的責任取決於 AWS 服務 您使用的 ，以及其他因素，包括資料的敏感度、組織的需求，以及適用的法律和法規。

主題

- 適用於 Ruby 的 AWS SDK 中的資料保護
- 適用於 Ruby 的 AWS SDK 的 Identity and Access Management
- 適用於 Ruby 的 AWS SDK 合規驗證
- 適用於 Ruby 的 AWS SDK 彈性
- 適用於 Ruby 的 AWS SDK 的 Infrastructure Security
- 在適用於 Ruby 的 AWS SDK 中強制執行最低 TLS 版本
- Amazon S3 加密用戶端遷移 (V1 至 V2)
- Amazon S3 加密用戶端遷移 (V2 到 V3)

適用於 Ruby 的 AWS SDK 中的資料保護

AWS 共同責任模型適用於 中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。 如需歐洲資料保護的詳細資訊，請參閱一般資料保護規則 (GDPR) 中心。

基於資料保護目的，建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。

- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 或使用主控台、API AWS CLI或其他 AWS 服務 AWS SDKs 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

適用於 Ruby 的 AWS SDK 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一種 Amazon Web Services (AWS) 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可以控制身分身分驗證 (已登入) 和授權 (具有許可) 以使用 AWS 服務資源。IAM 是 AWS 服務 您可以免費使用的 。

若要使用適用於 Ruby 的 AWS SDK 存取 AWS，您需要 AWS 帳戶和 AWS 登入資料。為提高 AWS 帳戶的安全性，建議您使用 IAM 使用者 提供存取登入資料，而不要使用 AWS 帳戶登入資料。

如需使用 IAM 的詳細資訊，請參閱 IAM。

如需概略了解 IAM 使用者以及它們對帳戶安全性的重要性，請參閱 Amazon Web Services 一般參考中的 AWS 安全登入資料。

AWS 適用於 Ruby 的 SDK 透過其支援的特定 Amazon Web Services (AWS) 服務遵循共同責任模型。如需 AWS 服務 安全性資訊，請參閱AWS 服務 安全性文件頁面AWS 服務 ，以及合規計劃在 AWS 合規工作範圍內的 。

適用於 Ruby 的 AWS SDK 合規驗證

AWS 適用於 Ruby 的 SDK 透過其支援的特定 Amazon Web Services (AWS) 服務遵循共同責任模型。如需 AWS 服務 安全性資訊，請參閱AWS 服務 安全性文件頁面AWS 服務 ，以及合規計劃在 AWS 合規工作範圍內的 。

Amazon Web Services (AWS) 服務的安全性和合規性由第三方稽核人員評估，作為多個 AWS 合規計劃的一部分。其中包括 SOC、PCI、FedRAMP、HIPAA 等。 AWS 提供合規計劃範圍內 Services AWS 服務 in 特定AWS 合規計劃範圍的經常更新清單。

可使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱以 AWS 成品下載報告。

如需 AWS 合規計劃的詳細資訊，請參閱AWS 合規計劃。

使用適用於 Ruby 的 AWS SDK 存取 時，您的合規責任 AWS 服務 取決於資料的敏感度、組織的合規目標，以及適用的法律和法規。如果您使用 AWS 服務 符合 HIPAA、PCI 或 FedRAMP 等標準，AWS 會提供資源來協助：

- 安全與合規快速入門指南 – 部署指南，討論架構考量，並提供在其中部署以安全為重心和以合規為重心的基準環境的步驟 AWS。
- HIPAA 合格服務參考 – 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- AWS 合規資源 – 工作手冊和指南的集合，可能適用於您的產業和據點。
- AWS Config – 評估資源組態是否符合內部實務、產業準則和法規的服務。
- AWS Security Hub – 內安全狀態的完整檢視 AWS ，可協助您檢查是否符合安全產業標準和最佳實務。

適用於 Ruby 的 AWS SDK 彈性

Amazon Web Services (AWS) 全球基礎設施是以 AWS 區域 和可用區域為基礎建置。

AWS 區域 提供多個實體分隔和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。

透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

AWS 適用於 Ruby 的 SDK 透過其支援的特定 Amazon Web Services (AWS) 服務遵循共同責任模型。如需 AWS 服務 安全性資訊，請參閱AWS 服務 安全性文件頁面AWS 服務 ，以及合規計劃在 AWS 合規工作範圍內的 。

適用於 Ruby 的 AWS SDK 的 Infrastructure Security

AWS 適用於 Ruby 的 SDK 透過其支援的特定 Amazon Web Services (AWS) 服務遵循共同責任模型。如需 AWS 服務 安全性資訊，請參閱AWS 服務 安全性文件頁面AWS 服務 ，以及合規計劃在 AWS 合規工作範圍內的 。

如需 AWS 安全程序的資訊，請參閱 AWS：安全程序概觀白皮書。

在適用於 Ruby 的 AWS SDK 中強制執行最低 TLS 版本

適用於 Ruby 的 AWS SDK 與 之間的通訊 AWS 會使用 Secure Sockets Layer (SSL) 或 Transport Layer Security (TLS) 進行保護。所有版本的 SSL 和早於 1.2 的 TLS 都有漏洞，可能會危及您與 通訊的安全性。 AWS因此，您應該確保使用適用於 Ruby 的 AWS SDK 搭配支援 TLS 1.2 版或更新版本的 Ruby 版本。

Ruby 使用 OpenSSL 程式庫來保護 HTTP 連線。透過系統套件管理員 (yum、 等） apt、官方安裝程式或 Ruby 管理員 (rbenv、RVM 等） 安裝的 Ruby 支援版本 (1.9.3 及更新版本） 通常會包含 OpenSSL 1.0.1 或更新版本，以支援 TLS 1.2。

當與支援版本的 Ruby 搭配 OpenSSL 1.0.1 或更新版本搭配使用時，適用於 Ruby 的 AWS SDK 偏好 TLS 1.2，並使用用戶端和伺服器支援的最新版本 SSL 或 TLS。這一律至少適用於 TLS 1 AWS 服務.2。（開發套件搭配 使用 Ruby `Net::HTTP`類別`use_ssl=true`。）

檢查 OpenSSL 版本

若要確保您安裝的 Ruby 是使用 OpenSSL 1.0.1 或更新版本，請輸入下列命令。

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

取得 OpenSSL `openssl` 版本的替代方法是直接查詢可執行檔。首先，使用以下命令找到適當的可執行檔。

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

輸出應該會`--with-openssl-dir=/path/to/openssl`指出 OpenSSL 安裝的位置。請記下此路徑。若要檢查 OpenSSL 的版本，請輸入下列命令。

```
cd /path/to/openssl
```

```
bin/openssl version
```

此後者方法可能無法用於 Ruby 的所有安裝。

升級 TLS 支援

如果您的 Ruby 安裝使用的 OpenSSL 版本早於 1.0.1，請使用您的系統套件管理員、Ruby 安裝程式或 Ruby 管理員升級 Ruby 或 OpenSSL 安裝，如 Ruby 安裝指南中所述。如果您要從來源安裝 Ruby，請先安裝最新的 OpenSSL，然後在執行 `--with-openssl-dir=/path/to/upgraded/openssl` 時傳遞 `./configure`。

Amazon S3 加密用戶端遷移 (V1 至 V2)

> **Note**
>
> 如果您使用的是 S3 加密用戶端的 V2，而且想要遷移至 V3，請參閱 Amazon S3 加密用戶端遷移 (V2 到 V3)。

本主題說明如何將應用程式從 Amazon Simple Storage Service (Amazon S3) 加密用戶端的第 1 版 (V1) 遷移到第 2 版 (V2)，並確保在整個遷移過程中的應用程式可用性。

遷移概觀

此遷移分為兩個階段：

1. 更新現有用戶端以讀取新格式。首先，將適用於 Ruby 的 AWS SDK 更新版本部署到您的應用程式。這將允許現有的 V1 加密用戶端解密新 V2 用戶端寫入的物件。如果您的應用程式使用多個 AWS SDKs，您必須分別升級每個 SDK。

2. 將加密和解密用戶端遷移至 V2。一旦所有 V1 加密用戶端都可以讀取新的格式，您就可以將現有的加密和解密用戶端遷移到各自的 V2 版本。

更新現有用戶端以讀取新格式

V2 加密用戶端使用舊版用戶端不支援的加密演算法。遷移的第一步是將 V1 解密用戶端更新為最新的 SDK 版本。完成此步驟後，應用程式的 V1 用戶端將能夠解密 V2 加密用戶端加密的物件。請參閱以下適用於 Ruby 的 AWS SDK 每個主要版本的詳細資訊。

更新適用於 Ruby 的 AWS SDK 第 3 版

第 3 版是適用於 Ruby 的 AWS SDK 最新版本。若要完成此遷移，您需要使用 1.76.0 版或更新版本的 `aws-sdk-s3` Gem 套件。

從命令列安裝

對於安裝 `aws-sdk-s3` Gem 的專案，請使用版本選項來驗證已安裝最低版本 1.76.0。

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

使用 Gemfile

對於使用 Gemfile 管理相依性的專案，請將 `aws-sdk-s3` Gem 的最低版本設定為 1.76.0。例如：

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. 修改 Gemfile。
2. 執行 `bundle update aws-sdk-s3`。若要驗證您的版本，請執行 `bundle info aws-sdk-s3`。

適用於 Ruby 的升級 AWS SDK 第 2 版

適用於 Ruby 的 AWS SDK 第 2 版將於 2021 年 11 月 21 日進入維護模式。若要完成此遷移，您需要使用 2.11.562 版或更新版本的 aws-sdk Gem 套件。

從命令列安裝

對於安裝 `aws-sdk` Gem 的專案，請從命令列使用版本選項來驗證已安裝最低版本 2.11.562。

```
gem install aws-sdk -v '>= 2.11.562'
```

使用 Gemfile

對於使用 Gemfile 管理相依性的專案，請將 `aws-sdk` Gem 的最低版本設定為 2.11.562。例如：

```
gem 'aws-sdk', '>= 2.11.562'
```

1. 修改 Gemfile。如果您有 Gemfile.lock 檔案，請刪除或更新它。

2. 執行 `bundle update aws-sdk`。若要驗證您的版本，請執行 `bundle info aws-sdk`。

將加密和解密用戶端遷移至 V2

將用戶端更新為讀取新的加密格式後，您可以將應用程式更新為 V2 加密和解密用戶端。下列步驟說明如何成功將程式碼從 V1 遷移至 V2。

在更新您的程式碼以使用 V2 加密用戶端之前，請確定您已遵循上述步驟，並使用 `aws-sdk-s3` Gem 2.11.562 版或更新版本。

> **Note**
>
> 使用 AES-GCM 解密時，請先將整個物件讀到結尾，再開始使用解密的資料。這是為了驗證物件在加密後尚未修改。

設定 V2 加密用戶端

EncryptionV2：：Client 需要額外的組態。如需詳細的組態資訊，請參閱 EncryptionV2：：Client 文件或本主題稍後提供的範例。

1. 金鑰包裝方法和內容加密演算法必須在用戶端建構時指定。建立新的 時`EncryptionV2::Client`，您需要提供 `key_wrap_schema`和 的值`content_encryption_schema`。

`key_wrap_schema` - 如果您使用的是 AWS KMS，這必須設定為 `:kms_context`。如果您使用對稱 (AES) 金鑰，則必須將其設定為 `:aes_gcm`。如果您使用非對稱 (RSA) 金鑰，則必須將其設定為 `:rsa_oaep_sha1`。

`content_encryption_schema` - 這必須設定為 ：aes_gcm_no_padding。

2. 必須在用戶端建構時指定 security_profile。建立新的 時`EncryptionV2::Client`，您需要提供 的值`security_profile`。security_profile 參數決定支援讀取使用舊版 V1 寫入的物件`Encryption::Client`。有兩個值：：v2 和 ：v2_and_legacy。若要支援遷移，請將`security_profile`設定為 ：v2_and_legacy。僅將 ：v2 用於新的應用程式開發。

3. AWS KMS key ID 預設會強制執行。在 V1 中，`kms_key_id`用於建立用戶端的`Encryption::Client`並未提供給 AWS KMS `Decrypt call`。 AWS KMS 可以從中繼

資料取得此資訊，並將其新增至對稱加密文字 Blob。在 V2 中，E`ncryptionV2：：Client` 會將 kms_key_id 傳遞給 AWS KMS Decrypt 呼叫，如果呼叫不符合用來加密物件的金鑰，則呼叫會失敗。如果您的程式碼先前倚賴未設定特定 kms_key_id，請在建立用戶端kms_key_id: :kms_allow_decrypt_with_any_cmk時設定 ，或在get_object呼叫kms_allow_decrypt_with_any_cmk: true時設定 。

範例：使用對稱 (AES) 金鑰

預遷移

```
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)
```

遷移後

```
client = Aws::S3::EncryptionV2::Client.new(
  encryption_key: rsa_key,
  key_wrap_schema: :rsa_oaep_sha1, # the key_wrap_schema must be rsa_oaep_sha1 for
 asymmetric keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
 )
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No changes
```

範例： AWS KMS 搭配 kms_key_id 使用

預遷移

```
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)
```

遷移後

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
```

```
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key) # No change
```

範例：使用 AWS KMS 而不搭配 kms_key_id

預遷移

```
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)
```

遷移後

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:
 true) # To allow decrypting with any cmk
```

遷移後替代方案

如果您只使用 S2 加密用戶端讀取和解密（絕不寫入和加密）物件，請使用此程式碼。

```
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
```

Amazon S3 加密用戶端遷移 (V2 到 V3)

> **Note**
>
> 如果您使用的是 S3 加密用戶端的 V1，您必須先遷移至 V2，才能遷移至 V3。如需從 V1 遷移至 V2 的說明Amazon S3 加密用戶端遷移 (V1 至 V2)，請參閱 。

本主題說明如何將應用程式從 Amazon Simple Storage Service (Amazon S3) 加密用戶端的第 2 版 (V2) 遷移到第 3 版 (V3)，並確保在整個遷移過程中的應用程式可用性。V3 推出具有金鑰承諾和承諾政策的 AES GCM，以增強安全性並防止資料金鑰竄改。

遷移概觀

Amazon S3 加密用戶端第 3 版推出具有金鑰承諾的 AES GCM，以提高安全性。這種新的加密演算法提供保護，防止資料金鑰遭到竄改，並確保加密資料的完整性。遷移至 V3 需要仔細規劃，以在整個過程中維持應用程式的可用性和資料可存取性。

此遷移分為兩個階段：

1. 更新現有用戶端以讀取新格式。首先，將適用於 Ruby 的 AWS SDK 更新版本部署到您的應用程式。這將允許現有的 V2 加密用戶端解密新 V3 用戶端寫入的物件。如果您的應用程式使用多個 AWS SDKs，您必須分別升級每個 SDK。

2. 將加密和解密用戶端遷移至 V3。一旦所有 V2 加密用戶端都可以讀取新的格式，您就可以將現有的加密和解密用戶端遷移到各自的 V3 版本。這包括設定承諾政策並更新您的程式碼，以使用新的用戶端組態選項。

如果您尚未從 V1 遷移至 V2，您必須先完成該遷移。如需從 V1 遷移至 V2 的詳細說明Amazon S3 加密用戶端遷移 (V1 至 V2)，請參閱 。

了解 V3 功能

Amazon S3 加密用戶端第 3 版推出兩個主要安全功能：承諾政策和具有金鑰承諾的 AES GCM。了解這些功能對於規劃遷移策略並確保加密資料的安全性至關重要。

承諾政策

承諾政策控制加密用戶端在加密和解密操作期間處理金鑰承諾的方式。金鑰承諾可確保加密的資料只能使用用來加密的確切金鑰進行解密，以防止特定類型的密碼編譯攻擊。

V3 加密用戶端支援三個承諾政策選項：

FORBID_ENCRYPT_ALLOW_DECRYPT

此政策在沒有金鑰承諾的情況下加密物件，並允許使用和不使用金鑰承諾來解密這兩個物件。

- 加密行為：使用與 V2 相同的演算法套件來加密物件，無需金鑰承諾。
- 解密行為：可以解密有或沒有金鑰承諾的加密物件。
- 安全性影響：此政策不會強制執行金鑰承諾，並可能允許竄改。使用此政策加密的物件不會受益於金鑰承諾的增強型安全保護。當您需要維持與 V2 加密行為的相容性時，請僅在遷移期間使用此政策。
- 版本相容性：使用此政策加密的物件可由 S3 加密用戶端的所有 V2 和 V3 實作讀取。 V3

REQUIRE_ENCRYPT_ALLOW_DECRYPT

此政策會使用金鑰承諾來加密物件，並允許使用和不使用金鑰承諾來解密這兩個物件。

- 加密行為：使用 AES GCM 搭配金鑰承諾，以金鑰承諾加密物件。
- 解密行為：可以解密有或沒有金鑰承諾的加密物件，提供回溯相容性。
- 安全影響：新物件受益於金鑰承諾保護，而沒有金鑰承諾的現有物件仍可讀取。這可在遷移期間提供安全與回溯相容性之間的平衡。
- 版本相容性：使用此政策加密的物件只能由 SV3 加密用戶端的 V3 和最新的 V2 實作讀取。 S3

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

此政策會加密具有金鑰承諾的物件，並僅允許解密使用金鑰承諾加密的物件。

- 加密行為：使用 AES GCM 搭配金鑰承諾，以金鑰承諾加密物件。
- 解密行為：只能解密以金鑰承諾加密的物件。嘗試在沒有金鑰承諾的情況下解密物件將會失敗。
- 安全性影響：此政策會強制執行所有操作的金鑰承諾，以提供最高層級的安全性。只有在所有物件都已透過金鑰承諾重新加密，且所有用戶端都已升級至 V3 之後，才能使用此政策。
- 版本相容性：使用此政策加密的物件只能由 SV3 加密用戶端的 V3 和最新的 V2 實作讀取。 S3 此政策也可防止讀取 V2 或 V1 用戶端加密的物件。

> **Note**
>
> 規劃遷移時，請從 開始REQUIRE_ENCRYPT_ALLOW_DECRYPT，以維持回溯相容性，同時獲得新物件金鑰承諾的安全優勢。只有在所有物件都重新加密且所有用戶端都升級至 V3 REQUIRE_ENCRYPT_REQUIRE_DECRYPT之後，才能移至 。

具有金鑰承諾的 AES GCM

具有金鑰承諾的 AES GCM (ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY) 是 V3 中引入的新加密演算法，可透過防止資料金鑰竄改來提供增強的安全性。了解此演算法的運作方式以及何時套用對於規劃遷移至關重要。

與先前演算法**ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY**的差異

舊版 S3 加密用戶端使用 AES CBC 或 AES GCM，無需金鑰承諾即可加密指令檔案中的資料金鑰。 會將密碼編譯承諾ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY新增至加密程序，將加密的資料繫結至特定金鑰。這可防止攻擊者竄改指令檔案中的加密資料金鑰，並導致用戶端使用不正確的金鑰解密資料。

如果沒有金鑰承諾，攻擊者可能會修改指示檔案中的加密資料金鑰，使其解密為不同的金鑰，進而可能允許未經授權的存取或資料損毀。 會確保加密的資料金鑰只能解密為加密期間使用的原始金鑰，以防止ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY此攻擊。

版本相容性

使用 加密的物件ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY只能由 SV3 加密用戶端的 V3 實作和包含支援讀取 V3 格式的特定 V2 轉換版本解密。 S3 沒有此轉換支援的 V2 用戶端無法解密使用加密的指令檔案ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY。

> **Warning**
>
> 使用 啟用加密 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY（ 使用 REQUIRE_ENCRYPT_ALLOW_DECRYPT或 REQUIRE_ENCRYPT_REQUIRE_DECRYPT承諾政策） 之前，請確保需要讀取加密物件的所有用戶端都已升級至 V3 或支援 V3 格式的轉換版本。如果任何沒有轉換支援的 V2 用戶端嘗試讀取使用 加密的物件ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY，解密將會失敗。

在遷移期間，您可以使用FORBID_ENCRYPT_ALLOW_DECRYPT承諾政策繼續加密，而無需 ，ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY同時仍然允許 V3 用戶端讀取使用金鑰承諾加密的物件。這提供安全的遷移路徑，您可以在其中先升級所有讀取器，然後切換到使用金鑰承諾加密。

更新現有用戶端以讀取新格式

V3 加密用戶端使用 V2 用戶端預設不支援的加密演算法和金鑰承諾功能。遷移的第一步是將您的 V2 解密用戶端更新為可讀取 V3 加密物件的適用於 Ruby 的 AWS SDK 版本。完成此步驟後，應用程式的 V2 用戶端將能夠解密由 V3 加密用戶端加密的物件。

若要讀取 V3 用戶端加密的物件（使用 REQUIRE_ENCRYPT_ALLOW_DECRYPT或 REQUIRE_ENCRYPT_REQUIRE_DECRYPT承諾政策的物件），您需要使用 1.93.0 版或更新版本的 aws-sdk-s3 Gem 套件。此版本包含支援解密使用 AES GCM 與金鑰承諾加密的物件。

從命令列安裝

對於從命令列安裝 aws-sdk-s3 Gem 的專案，請使用版本選項來驗證已安裝最低版本 1.208.0。

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

使用 Gemfile

對於使用 Gemfile 管理相依性的專案，請將 aws-sdk-s3 Gem 的最低版本設定為 1.208.0。例如：

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. 修改 Gemfile 以指定最低版本。
2. 執行 bundle update aws-sdk-s3以更新 Gem。
3. 若要驗證您的版本，請執行 bundle info aws-sdk-s3。

> **Note**
>
> 更新至最新版本後，您現有的 V2 加密用戶端將能夠解密由 V3 用戶端加密的物件。不過，它們將繼續使用 V2 演算法加密新物件，直到您將其遷移至 V3，如下一節所述。

將加密和解密用戶端遷移至 V3

將用戶端更新為讀取新的加密格式後，您可以將應用程式更新為 V3 加密和解密用戶端。下列步驟說明如何成功將程式碼從 V2 遷移至 V3。

在更新您的程式碼以使用 V3 加密用戶端之前，請確定您已遵循上述步驟，並使用 `aws-sdk-s3` Gem 1.93.0 版或更新版本。

> **Note**
>
> 使用 AES-GCM 解密時，請先將整個物件讀到結尾，再開始使用解密的資料。這是為了驗證物件在加密後尚未修改。

設定 V3 用戶端

V3 加密用戶端推出新的組態選項，可控制金鑰承諾行為和回溯相容性。了解這些選項對於成功遷移至關重要。

commitment_policy

`commitment_policy` 參數控制加密用戶端在加密和解密操作期間處理金鑰承諾的方式。這是 V3 用戶端最重要的組態選項。

- `:require_encrypt_allow_decrypt` - 使用金鑰承諾加密新物件，並允許解密具有或不具有金鑰承諾的物件。這是建議的遷移設定，因為它可為新物件提供增強的安全性，同時保持與現有 V2 物件的回溯相容性。
- `:forbid_encrypt_allow_decrypt` - 加密沒有金鑰承諾的新物件（使用 V2 演算法），並允許解密有或沒有金鑰承諾的物件。只有在您需要在遷移期間維護 V2 加密行為時，例如某些用戶端尚無法讀取 V3 加密物件時，才使用此設定。
- `:require_encrypt_require_decrypt` - 使用金鑰承諾加密新物件，並僅允許使用金鑰承諾加密的物件進行解密。只有在所有物件都已透過金鑰承諾重新加密，且所有用戶端都已升級至 V3 之後，才能使用此設定。

security_profile

`security_profile` 參數決定對讀取較舊加密用戶端版本所寫入物件的支援。此參數對於在遷移期間維持回溯相容性至關重要。

- :v3_and_legacy - 允許 V3 用戶端解密 V1 和 V2 加密用戶端加密的物件。在遷移期間使用此設定，以確保您的 V3 用戶端可以讀取所有現有的加密物件。
- :v3 - 允許 V3 用戶端僅解密 V2 加密用戶端加密的物件。如果您已將所有 V1 物件遷移至 V2 格式，請使用此設定。
- 如果未指定，用戶端只會解密 V3 用戶端加密的物件。僅將此用於不存在舊版物件的新應用程式開發。

envelope_location

envelope_location 參數會決定加密中繼資料（包括加密的資料金鑰）的存放位置。此參數會影響哪些物件受到具有金鑰承諾的 AES GCM 保護。

- :metadata（預設）- 將加密中繼資料存放在 S3 物件的中繼資料標頭中。這是預設行為，建議大多數使用案例使用。使用中繼資料儲存體時，不適用具有金鑰承諾的 AES GCM。
- :instruction_file - 將加密中繼資料存放在具有可設定尾碼的個別 S3 物件（指令檔案）中。使用指示檔案時，AES GCM 與金鑰承諾可防止加密的資料金鑰遭到竄改。如果您需要資料金鑰本身的金鑰承諾提供的額外安全性，請使用此設定。

使用 時:instruction_file，您可以選擇指定 instruction_file_suffix 參數，以自訂用於指令檔案物件的尾碼。預設尾碼為 .instruction。

何時使用每個組態選項

在遷移期間，請遵循以下建議的組態策略：

1. 初始遷移：設定 commitment_policy: :require_encrypt_allow_decrypt和 security_profile: :v3_and_legacy。這可讓您的 V3 用戶端使用金鑰承諾加密新物件，同時仍然能夠解密所有現有的 V1 和 V2 物件。
2. 升級所有用戶端之後：繼續使用 commitment_policy: :require_encrypt_allow_decrypt和 ，security_profile: :v3_and_ 到您重新加密所有需要金鑰承諾保護的物件為止。
3. 完整 V3 強制執行：只有在所有物件都已使用金鑰承諾重新加密，而且您不再需要讀取 V1/V2 物件之後，您可以選擇切換到 commitment_policy: :require_encrypt_require_decrypt 並移除 security_profile 參數（如果 V2 物件仍然存在:v2，則將其設定為)。

對於 envelope_location，除非您有特定的變更原因，否則請繼續使用現有的儲存方法 (:metadata 或 :instruction_file)。如果您目前正在使用中繼資料儲存體，並希望對資料金鑰具有金鑰承諾的 AES GCM 具有額外的安全性，您可以切換到 :instruction_file，但請注意，這將需要更新讀取這些物件的所有用戶端。

將加密和解密用戶端遷移至 V3

將用戶端更新為讀取新的加密格式後，您可以將應用程式更新為 V3 加密和解密用戶端。下列範例示範如何成功將程式碼從 V2 遷移至 V3。

使用 V3 加密用戶端

預遷移 (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
```

在遷移期間 (具有回溯相容性的 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
```

遷移後 (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 的關鍵差異是新增 commitment_policy 參數。將其設定為 :require_encrypt_require_decrypt 可確保使用金鑰承諾加密新物件，而且用戶端只會解密使用金鑰承諾加密的物件，從而增強資料金鑰竄改的安全性。

put_object 呼叫本身保持不變。所有安全性增強功能都是在用戶端層級設定。

其他範例

本節提供特定遷移案例的其他範例，以及在 V2 到 V3 遷移期間可能有用的組態選項。

指示檔案與中繼資料儲存

S3 加密用戶端可以將加密中繼資料（包括加密的資料金鑰）存放在兩個不同的位置：S3 物件的中繼資料標頭或單獨的指令檔案中。儲存方法的選擇會影響哪些物件受益於具有金鑰承諾保護的 AES GCM。

中繼資料儲存（預設）

根據預設，加密用戶端會將加密中繼資料存放在 S3 物件的中繼資料標頭中。這是大多數使用案例的建議方法，因為它會將加密中繼資料與 物件一起保留，而且不需要管理個別的指令檔案物件。

```
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')
```

使用中繼資料儲存體時，具有金鑰承諾的 AES GCM 不適用於加密的資料金鑰。不過，使用 `commitment_policy: :require_encrypt_allow_decrypt`或 時，內容加密仍受益於金鑰承諾`:require_encrypt_require_decrypt`。

指令檔案儲存

或者，您可以設定加密用戶端，將加密中繼資料存放在稱為 指令檔案的個別 S3 物件中。搭配 V3 使用指令檔案時，加密的資料金鑰受到 AES GCM 與金鑰承諾的保護，可提供額外的安全性，避免資料金鑰遭到竄改。

```
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 時`envelope_location: :instruction_file`，加密用戶端會建立兩個 S3 物件：

1. 加密的資料物件（例如 `my-object`)
2. 包含加密中繼資料的指示檔案（例如 `my-object.instruction`)

`instruction_file_suffix` 參數可讓您自訂用於指令檔案的尾碼。預設值為 `.instruction`。

何時使用每種儲存方法

- 針對大多數案例使用中繼資料儲存。其可簡化物件管理，因為加密中繼資料會隨著物件移動。
- 當需要考慮物件中繼資料大小，或您需要將加密中繼資料與加密物件分開時，請使用指令檔案儲存。請注意，使用指令檔案需要管理兩個 S3 物件（加密的物件及其指令檔案），而不是一個。

> **Warning**
>
> 如果您從中繼資料儲存變更為指令檔案儲存（反之亦然），使用新儲存方法設定的用戶端將無法讀取使用舊儲存方法加密的現有物件。仔細規劃您的儲存方法，並保持整個應用程式的一致性。

文件歷史記錄

下表說明本指南中的重要變更。如需獲得此文件更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
內容重組	更新目錄和內容組織，以更符合其他 AWS SDKs。	2025 年 3 月 29 日
可觀測性	新增 Ruby 可觀測性的相關資訊。	2025 年 1 月 24 日
一般更新	將最低必要 Ruby 版本更新為 v2.5。已更新資源連結。	2024 年 11 月 13 日
目錄和引導式範例	移除引導式範例，以延遲到更全面的程式碼範例儲存庫。	2024 年 7 月 10 日
目錄	更新了目錄，讓程式碼範例更容易存取。	2023 年 6 月 1 日
IAM 最佳實務更新	更新了指南以符合 IAM 最佳實務。如需更多詳細資訊，請參閱 IAM 中的安全最佳實務。入門的更新。	2023 年 5 月 8 日
一般更新	更新相關外部資源的歡迎頁面。也更新了 v2.3 的最低必要 Ruby 版本。更新 AWS Key Management Service 章節以反映術語更新。為了清楚起見，已更新 REPL 公用程式的使用資訊。	2022 年 8 月 8 日
修正中斷的連結	修正中斷的範例連結。已移除備援提示和提示頁面；重新導向至 Amazon EC2 範例內容。包含程式碼範例儲存庫中	2022 年 8 月 3 日

	GitHub 上提供的程式碼範例清單。	
SDK 指標	已移除有關為企業支援啟用 SDK 指標的資訊，該指標已棄用。	2022 年 1 月 28 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。