Guía para desarrolladores

AWS SDKpara Ruby

AWS SDKpara Ruby: Guía para desarrolladores

Table of Contents

¿Qué es AWS SDK para Ruby?

Le damos la bienvenida a la guía para desarrolladores de AWS SDK para Ruby. AWS SDK para Ruby proporciona bibliotecas de soporte para casi todos los Servicios de AWS, incluidos Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2) y Amazon DynamoDB.

La Guía para desarrolladores de AWS SDK para Ruby proporciona información sobre cómo instalar, configurar y usar AWS SDK para Ruby para crear aplicaciones Ruby que usen Servicios de AWS.

Introducción a AWS SDK para Ruby

Documentación y recursos adicionales

Más recursos para desarrolladores de AWS SDK para Ruby; se encuentran en:

- Guía de referencia de las herramientas y los SDK de AWS: incluye configuraciones, funciones y otros conceptos básicos comunes a los SDK de AWS.
- Referencia de la API de AWS SDK for Ruby versión 3
- Repositorio de ejemplos de código AWS en GitHub
- RubyGems.org: la última versión del SDK está modularizada en gemas específicas de cada servicio, disponibles aquí
  - Servicios compatibles: enumera todas las gemas compatibles con AWS SDK para Ruby
- AWSFuente de SDK para Ruby en GitHub:
  - Fuente y README
  - Registros de cambios bajo cada gema
  - Pasar de v2 a v3
  - Problemas
  - Notas sobre la actualización principal
- Blog de desarrolladores
- Canal de Gitter
- @awsforruby en Twitter

Implementación en la nube de AWS

Puede utilizar los Servicios de AWS, por ejemplo, AWS Elastic Beanstalk, AWS OpsWorks y AWS CodeDeploy para implementar su aplicación en la nube de AWS. Para implementar aplicaciones Ruby con Elastic Beanstalk, consulte Implementación de aplicaciones de Elastic Beanstalk en Using con la CLI de EB y Git en la Guía del desarrollador de AWS Elastic Beanstalk. Para implementar una aplicación de Ruby on Rails con AWS OpsWorks, consulte Implementación de aplicaciones Ruby on Rails en AWS OpsWorks. Para obtener información general sobre los servicios de implementación de AWS, consulte Información general de las opciones de implementación en AWAWS.

Mantenimiento y compatibilidad de las versiones principales del SDK

Para obtener información sobre el mantenimiento y la compatibilidad con las principales versiones del SDK y sus dependencias subyacentes, consulte lo siguiente en la Guía de Referencia de SDK y herramientas de AWS:

- Política de mantenimiento de SDK y herramientas AWS
- Matriz de compatibilidad para versiones de SDK y herramientas AWS

Introducción a AWS SDK para Ruby

Aprenda a instalar, configurar y usar el SDK para crear una aplicación Ruby para acceder a un recurso de AWS mediante programación.

Temas

Autenticación de SDK con AWS

Debe establecer cómo se autentica el código con AWS cuando se desarrolla con Servicios de AWS. Puede configurar el acceso a los recursos de AWS mediante programación de diferentes maneras, según el entorno y el acceso a AWS disponibles.

Para elegir el método de autenticación y configurarlo para el SDK, consulte Autenticación y acceso en la Guía de referencia de las herramientas y los SDK de AWS.

Recomendamos que los nuevos usuarios que desarrollen a nivel local y no dispongan de un método de autenticación por parte de su empresa configuren AWS IAM Identity Center. Este método incluye la instalación de AWS CLI para facilitar la configuración y para iniciar sesión con regularidad en el portal de acceso de AWS. Si elige este método, su entorno debería contener los siguientes elementos después de completar el procedimiento de autenticación del IAM Identity Center descrito en la Guía de referencia de las herramientas y los SDK de AWS:

- La AWS CLI, que se utiliza para iniciar una sesión en el portal de acceso a AWS antes de ejecutar la aplicación.
- Un archivo `config` compartido de AWS que tiene un perfil `[default]` con un conjunto de valores de configuración a los que se puede hacer referencia desde el SDK. Para encontrar la ubicación de este archivo, consulte Ubicación de los archivos compartidos en la Guía de referencia de herramientas y AWS SDK.
- El archivo `config` compartido establece la configuración de `region`. Esto establece la Región de AWS predeterminada que el SDK usa para las solicitudes de AWS. Esta región se usa para las solicitudes de servicio del SDK que no tienen especificadas una región.

- El SDK utiliza la configuración de proveedor de token de SSO del perfil para adquirir las credenciales antes de enviar las solicitudes a AWS. El valor `sso_role_name`, que es un rol de IAM conectado a un conjunto de permisos del Centro de identidades de IAM, permite el acceso a los Servicios de AWS utilizados en la aplicación.

  El siguiente archivo `config` de ejemplo muestra la configuración de un perfil predeterminado con el proveedor de token de SSO. La configuración `sso_session` del perfil hace referencia a la sección llamada `sso-session`. La sección `sso-session` contiene la configuración para iniciar una sesión en el portal de acceso a AWS.

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

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

AWS SDK para Ruby no necesita añadir paquetes adicionales (por ejemplo, `SSO` y `SSOOIDC`) a la aplicación para utilizar la autenticación del IAM Identity Center.

Iniciar una sesión en el portal de acceso a AWS

Antes de ejecutar una aplicación que accede a Servicios de AWS, necesita una sesión activa en el portal de acceso a AWS para que el SDK utilice la autenticación del IAM Identity Center para resolver las credenciales. En función de la duración de las sesiones configuradas, el acceso terminará por caducar y el SDK detectará un error de autenticación. Para iniciar sesión en el portal de acceso a AWS, ejecute el siguiente comando en la AWS CLI.

```
aws sso login
```

Si sigue la guía y utiliza una configuración de perfil predeterminada, no necesita llamar al comando con una opción `--profile`. Si la configuración del proveedor de token de SSO utiliza un perfil con nombre, el comando es `aws sso login --profile named-profile`.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente comando de AWS CLI.

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

Si su sesión está activa, la respuesta a este comando debe indicar la cuenta y el conjunto de permisos del IAM Identity Center configurados en el archivo `config` compartido.

> **Note**
>
> Si ya tiene una sesión activa en el portal de acceso a AWS y ejecuta `aws sso login`, no tendrá que proporcionar credenciales.
> Es posible que el proceso de inicio de sesión le permita el acceso de la AWS CLI a los datos. Como AWS CLI se ha creado con el SDK para Python, los mensajes de permiso pueden contener variaciones del nombre `botocore`.

Información adicional de autenticación

Los usuarios humanos, que también reciben el nombre de identidades humanas, son las personas, los administradores, los desarrolladores, los operadores y los consumidores de las aplicaciones. Deben tener una identidad para acceder a los entornos y aplicaciones de AWS. Los usuarios humanos que son miembros de su organización (es decir, usted, el desarrollador) se conocen como identidades de personal.

Utilice credenciales temporales al acceder a AWS. Puede utilizar un proveedor de identidades para los usuarios humanos para proporcionar acceso federado a las cuentas de AWS asumiendo roles, que proporcionan credenciales temporales. Si desea administrar el acceso de manera centralizada, se recomienda utilizar AWS IAM Identity Center (IAM Identity Center) para administrar el acceso a las cuentas y los permisos de esas cuentas. Para obtener más alternativas, consulte lo siguiente:

- Para obtener más información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.
- Para crear credenciales de AWS de corta duración, consulte Credenciales de seguridad temporales en la Guía del usuario de IAM.
- Para obtener más información sobre otros proveedores de credenciales de AWS SDK para Ruby, consulte los proveedores de credenciales estandarizados en la Guía de referencia de las herramientas y los SDK de AWS.

Instalar AWS SDK para Ruby

Esta sección incluye los requisitos previos y las instrucciones de instalación de AWS SDK para Ruby.

Requisitos previos

Antes de utilizar AWS SDK para Ruby debe autenticarse con AWS. Para obtener información acerca de la configuración de la autenticación, consulte Autenticación de SDK con AWS.

Instalación del SDK

Puede instalar AWS SDK para Ruby como lo haría con cualquier gema de Ruby. Las gemas están disponibles en RubyGems. AWS SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. La instalación de toda la gema `aws-sdk` es larga y puede llevar más de una hora.

Le recomendamos que instale solo las gemas para los Servicios de AWS que utilice. Su denominación sigue el modelo `aws-sdk-`*`service_abbreviation`* y la lista completa se encuentra en la tabla Servicios compatibles del archivo README de AWS SDK para Ruby. Por ejemplo, la gema para interactuar con el servicio Amazon S3 está disponible directamente en `aws-sdk-s3`.

Administrador de versiones de Ruby

En lugar de usar el Ruby del sistema, recomendamos usar un administrador de versiones de Ruby como el siguiente:

- RVM
- chruby
- rbenv

Por ejemplo, si utiliza un sistema operativo Amazon Linux 2, puede usar los siguientes comandos para actualizar RVM, enumerar las versiones de Ruby disponibles y, a continuación, elegir la versión que desea usar para el desarrollo con AWS SDK para Ruby. La versión mínima de Ruby requerida es 2.3.

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

```
$ rvm --default use 3.1.3
```

Bundler

Si utiliza Bundler, los siguientes comandos instalan la gema AWS SDK para Ruby para Amazon S3:

1. Instale Bundler y cree el archivo `Gemfile`:

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

2. Abra el archivo `Gemfile` que ha creado y añada una línea de `gem` para cada gema de servicio de AWS que utilice su código. Para seguir con el ejemplo de Amazon S3, agregue la siguiente línea de texto al final del archivo:

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

3. Guarde el archivo Gemfile.
4. Instale las dependencias especificadas en su archivo `Gemfile`:

   ```
   $ bundle install
   ```

Tutorial de Hello World para AWS SDK para Ruby

Descubra Amazon S3 utilizando AWS SDK para Ruby. En el siguiente ejemplo se muestra una lista de sus buckets de Amazon S3.

Escriba el código

Copie y pegue el siguiente código en un nuevo archivo de origen. Nombre el archivo `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 SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. Una vez instalada la gema, la instrucción de `require` que aparece en la parte superior del archivo fuente de Ruby importa las clases y los métodos del SDK de AWS para el servicio Amazon S3. Para obtener una lista completa de las gemas de los servicios de AWS disponibles, consulte la tabla Servicios compatibles del archivo README de AWS SDK para Ruby.

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

Ejecución del programa

Abra un comando para ejecutar su programa Ruby. La sintaxis de comando habitual para ejecutar un programa Ruby es:

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

Este ejemplo de código no utiliza argumentos. Para ejecutar este código, introduzca el siguiente comando del sistema:

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

Nota para usuarios de Windows

Cuando utiliza certificados SSL en Windows y ejecuta el código Ruby, podría ver un error similar al siguiente.

```
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'
...
```

Para solucionar este problema, añada la siguiente línea a su archivo de código de Ruby, en algún lugar antes de la primera llamada a AWS.

```
Aws.use_bundled_cert!
```

Tenga en cuenta que si solo utiliza la gema `aws-sdk-s3` en su programa de Ruby, también tendrá que añadir la gema `aws-sdk-core` para utilizar el certificado agrupado.

Siguientes pasos

Para probar muchas otras operaciones de Amazon S3, consulte el repositorio de ejemplos de AWS código en GitHub.

AWS Cloud9 Úselo con AWS SDK para Ruby

AWS Cloud9 es un entorno de desarrollo integrado basado en la web (IDE) que contiene un conjunto de herramientas que se utilizan para codificar, compilar, ejecutar, probar, depurar y publicar software

en la nube. Puedes usarlo AWS Cloud9 con Ruby AWS SDK para escribir y ejecutar tu código de Ruby mediante un navegador. AWS Cloud9 incluye herramientas como un editor de código y un terminal. Como AWS Cloud9 IDE está basado en la nube, puede trabajar en sus proyectos desde la oficina, el hogar o cualquier lugar mediante una máquina conectada a Internet. Para obtener información general al respecto AWS Cloud9, consulte la Guía del AWS Cloud9 usuario.

Siga estas instrucciones para AWS Cloud9 configurarlo AWS SDK para Ruby:

- Paso 1: Configura tu Cuenta de AWS uso AWS Cloud9
- Paso 2: Configurar su entorno de desarrollo de AWS Cloud9
- Paso 3: Configura el AWS SDK para Ruby
- Paso 4: Descargar el código de ejemplo
- Paso 5: Ejecutar el código de ejemplo

Paso 1: Configura tu Cuenta de AWS uso AWS Cloud9

Para usarlo AWS Cloud9, inicia sesión en la AWS Cloud9 consola desde AWS Management Console.

> **Note**
>
> Si la utiliza AWS IAM Identity Center para autenticarse, es posible que necesite añadir el permiso necesario a la política asociada `iam:ListInstanceProfilesForRole` al usuario en la IAM consola.

Para configurar una IAM entidad en tu AWS cuenta para acceder a la AWS Cloud9 consola AWS Cloud9 e iniciar sesión en ella, consulta Configuración de equipo AWS Cloud9 en la Guía del AWS Cloud9 usuario.

Paso 2: Configurar su entorno de desarrollo de AWS Cloud9

Después de iniciar sesión en la AWS Cloud9 consola, utilícela para crear un entorno de AWS Cloud9 desarrollo. Tras crear el entorno, AWS Cloud9 abre el IDE para ese entorno.

Para obtener más información, consulte cómo crear un entorno en AWS Cloud9 en la guía del usuario de AWS Cloud9 .

> **Note**
>
> Al crear el entorno en la consola por primera vez, le recomendamos que elija la opción Crear una nueva instancia para el entorno (EC2). Esta opción indica AWS Cloud9 que hay que crear un entorno, lanzar una EC2 instancia de Amazon y, a continuación, conectar la nueva instancia al nuevo entorno. Esta es la forma más rápida de empezar a utilizarla AWS Cloud9.

Si el terminal aún no está abierto en elIDE, ábrelo. En la barra de menús delIDE, selecciona Ventana, Nueva terminal. Puede utilizar la ventana de terminal para instalar herramientas y crear sus aplicaciones.

Paso 3: Configura el AWS SDK para Ruby

Después de AWS Cloud9 abrir el IDE para su entorno de desarrollo, utilice la ventana de terminal para configurar el AWS SDK para Ruby en su entorno.

Puedes instalar AWS SDK para Ruby como lo harías con cualquier gema de Ruby. Las gemas están disponibles en RubyGems. El AWS SDK para Ruby está diseñado para ser modular y está separado por Servicio de AWS. La instalación de toda la gema `aws-sdk` es larga y puede llevar más de una hora.

Recomendamos instalar solo las gemas Servicios de AWS que utilice. Tienen el mismo nombre `aws-sdk-`*`service_abbreviation`* y la lista completa se encuentra en la tabla de servicios compatibles del AWS SDK README archivo Ruby. Por ejemplo, la gema para interactuar con el servicio Amazon S3 está disponible directamente en `aws-sdk-s3`.

Administrador de versiones de Ruby

En lugar de usar el Ruby del sistema, recomendamos usar un administrador de versiones de Ruby como el siguiente:

- RVM
- chruby
- rbenv

Por ejemplo, si utilizas un sistema operativo Amazon Linux 2, puedes usar los siguientes comandos para actualizarRVM, enumerar las versiones de Ruby disponibles y, a continuación, elegir la versión

que quieres usar AWS SDK para el desarrollo con Ruby. La versión mínima de Ruby requerida es 2.3.

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

Bundler

Si utilizas Bundler, los siguientes comandos instalan la gema AWS SDK for Ruby para Amazon S3:

1. Instale Bundler y cree el archivo `Gemfile`:

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

2. Abre la gema creada `Gemfile` y añade una `gem` línea para cada gema AWS de servicio que vaya a utilizar tu código. Para seguir con el ejemplo de Amazon S3, agregue la siguiente línea de texto al final del archivo:

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

3. Guarde el archivo Gemfile.
4. Instale las dependencias especificadas en su `Gemfile`:

   ```
   $ bundle install
   ```

Paso 4: Descargar el código de ejemplo

Usa la ventana del terminal para descargar el AWS SDK código de ejemplo de Ruby al entorno de AWS Cloud9 desarrollo.

Para descargar una copia de todos los ejemplos de código utilizados en la AWS SDK documentación oficial en el directorio raíz de su entorno, ejecute el siguiente comando:

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

Los ejemplos de código AWS SDK para Ruby se encuentran en el `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby` directorio, donde `ENVIRONMENT_NAME` aparece el nombre de su entorno de desarrollo.

Para seguir con un ejemplo de Amazon S3, le recomendamos empezar con un ejemplo de código de `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby/example_code/s3/bucket_list.rb`. Utilice la ventana de terminal para acceder al directorio de `s3` y enumerar los archivos.

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

Para abrir el archivo AWS Cloud9, puede hacer clic `bucket_list.rb` directamente en él en la ventana del terminal.

Paso 5: Ejecutar el código de ejemplo

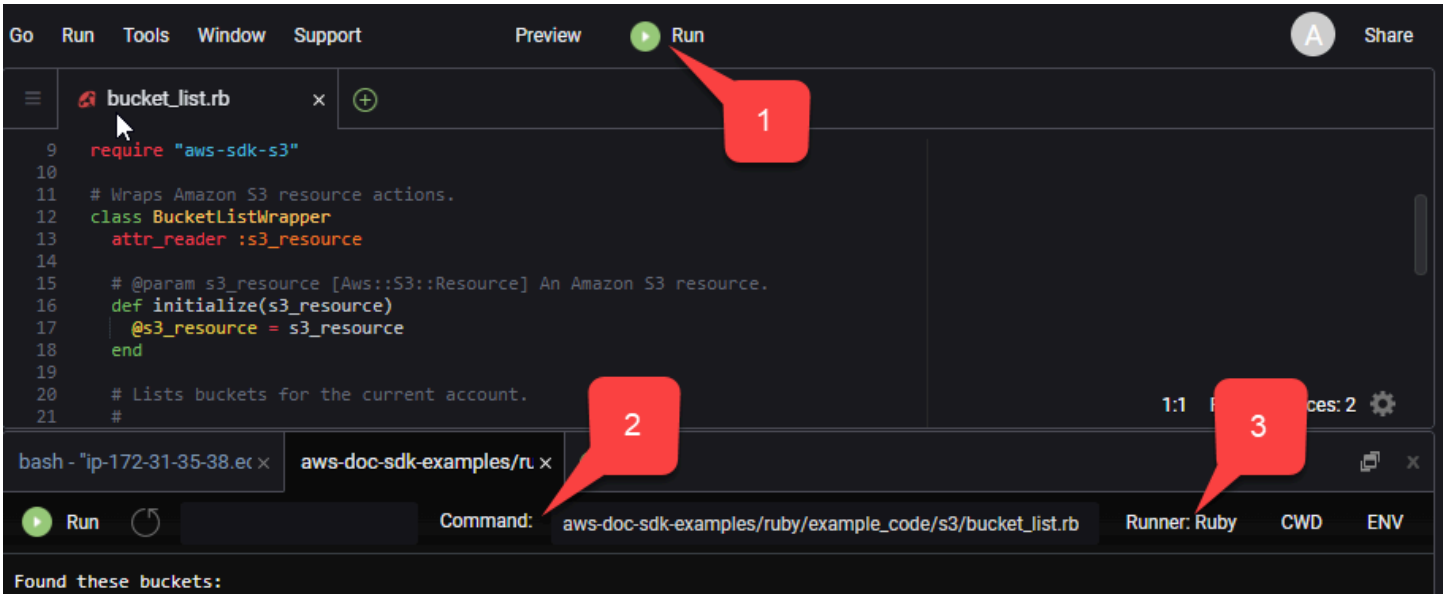
Para ejecutar código en su entorno de AWS Cloud9 desarrollo, pulse el botón Ejecutar en la barra de menú superior. AWS Cloud9 detectará automáticamente la extensión del `.rb` archivo y utilizará el ejecutor Ruby para ejecutar el código. Para obtener más información sobre cómo ejecutar código en AWS Cloud9, consulte Ejecute su código en la Guía del AWS Cloud9 usuario.

En la siguiente captura de pantalla, observe estas áreas básicas:

- 1: Botón Run (Ejecutar). El botón Run (Ejecutar) se encuentra en la barra de menú superior. Se abrirá una nueva pestaña para ver los resultados.

  > **Note**
  >
  > También puede crear nuevas configuraciones de ejecución manualmente. En la barra de menú, elija Run (Ejecutar), Run Configurations (Configuraciones de ejecución), New Run Configuration (Nueva configuración de ejecución).

- 2: Comando. AWS Cloud9 rellena el cuadro de texto del comando con la ruta y el nombre del archivo que está ejecutando. Si el código espera que se le pasen parámetros de línea de comandos, puede añadirlos a la línea de comandos del mismo modo que lo haría al ejecutar el código a través de una ventana de terminal.
- 3: Corredor. AWS Cloud9 detecta que la extensión de tu archivo es `.rb` y selecciona Ruby Runner para ejecutar tu código.



En la pestaña se muestra cualquier resultado generado a partir del código en ejecución.

Para probar muchas otras operaciones de Amazon S3, consulte el repositorio de ejemplos de AWS código en GitHub.

Configura el AWS SDK para Ruby

Aprenda a configurar AWS SDK para Ruby. Debe establecer cómo se autentica el código con AWS cuando desarrolla con Servicios de AWS. También debes configurar Región de AWS lo que quieres usar.

Cadena de proveedores de credenciales

Todos SDKs tienen una serie de lugares (o fuentes) que consultan para obtener credenciales válidas que puedan utilizarlas para realizar una solicitud a un Servicio de AWS. Una vez que se encuentran las credenciales válidas, se detiene la búsqueda. Esta búsqueda sistemática se denomina cadena predeterminada de proveedores de credenciales.

Para cada paso de la cadena, hay diferentes maneras de establecer los valores. La configuración de los valores directamente en el código siempre tiene prioridad, seguida de la configuración como variables de entorno y, por último, en el AWS `config` archivo compartido. Para obtener más información, consulte Prioridad de los ajustes en la Guía de referencia de herramientas AWS SDKs y herramientas.

La guía de referencia de AWS SDKs and Tools contiene información sobre los ajustes de SDK configuración utilizados por todos AWS SDKs y por. AWS CLI Para obtener más información sobre cómo configurarlos SDK a través del AWS `config` archivo compartido, consulte Archivos de credenciales y configuración compartidos. Para obtener más información sobre cómo configurar las variables de entorno SDK mediante la configuración, consulte Compatibilidad con variables de entorno.

Para autenticarse AWS, AWS SDK para Ruby comprueba los proveedores de credenciales en el orden que se indica en la siguiente tabla.

Proveedor de credenciales por prioridad	AWS SDKsy la guía de referencia de herramientas	AWS SDK for Ruby APIRefere ncia
AWS claves de acceso (credenciales temporales y de larga duración)	AWS claves de acceso	`Aws::Credentials` `Aws::SharedCredent ials`

Proveedor de credenciales por prioridad	AWS SDKsy la guía de referencia de herramientas	AWS SDK for Ruby APIReferencia
token de identidad web de AWS Security Token Service (AWS STS)	Asumir el rol de proveedor de credenciales Uso de `role_arn`, `role_session_name` y `web_identity_token_file`	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center. En esta guía, consulte Autenticación de SDK con AWS.	IAMProveedor de credenciales de Identity Center	`Aws::SSOCredentials`
Proveedor de entidades de confianza (como `AWS_ROLE_ARN`). En esta guía, consulte Crear un token de AWS STS acceso.	Asumir el rol de proveedor de credenciales Uso de `role_arn` y `role_session_name`	`Aws::AssumeRoleCredentials`
Proveedor de credenciales de proceso	Proveedor de credenciales de proceso	`Aws::ProcessCredentials`
Credenciales de Amazon Elastic Container Service (AmazonECS)	Proveedor de credenciales de contenedor	`Aws::ECSCredentials`
IMDSCredenciales de perfil de instancia de Amazon Elastic Compute Cloud (AmazonEC2) (proveedor de credenciales)	IMDSproveedor de credenciales	`Aws::InstanceProfileCredentials`

Si `AWS_SDK_CONFIG_OPT_OUT` se establece la variable de entorno AWS SDK for Ruby, el AWS `config` archivo compartido, normalmente at `~/.aws/config`, no se analizará en busca de credenciales.

Si siguió el enfoque recomendado para los nuevos usuarios para empezar, configuró la autenticación AWS IAM Identity Center en Autenticación de SDK con AWS del tema Introducción. Otros métodos de autenticación son útiles en diferentes situaciones. Para evitar riesgos de seguridad, recomendamos utilizar siempre credenciales a corto plazo. Para conocer otros procedimientos de métodos de autenticación, consulte Autenticación y acceso en la Guía de referencia de herramientas AWS SDKs y herramientas.

Crear un token de AWS STS acceso

Asumir un rol implica usar un conjunto de credenciales de seguridad temporales que puede usar para acceder a AWS recursos a los que normalmente no tendría acceso. Las credenciales temporales incluyen un ID de clave de acceso, una clave de acceso secreta y un token de seguridad. Puede usar el método `Aws::AssumeRoleCredentials` para crear un token de acceso AWS Security Token Service (AWS STS).

El siguiente ejemplo utiliza un token de acceso para crear un objeto de cliente de Amazon S3, donde `linked::account::arn` es el nombre del recurso de Amazon (ARN) del rol que se va a asumir y `session-name` es un identificador de la sesión del rol asumido.

```
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)
```

Para obtener más información sobre cómo `role_arn` configurarlos o `role_session_name` sobre cómo configurarlos utilizando el AWS `config` archivo compartido en su lugar, consulte Asumir el rol de proveedor de credenciales en la Guía de referencia de herramientas AWS SDKs y herramientas.

Configuración de una región

Debe configurar una región cuando se utiliza la mayoría de los Servicios de AWS. AWS SDKPara Ruby, busca una región en el siguiente orden:

1. Configuración de la región en un objeto de cliente o recurso
2. Configuración de la región mediante `Aws.config`

3. Configuración de la región mediante variables de entorno
4. Configuración de la región mediante el archivo compartido `config`

Para obtener más información sobre la `region` configuración, consulte Región de AWSla Guía de referencia de herramientas AWS SDKs y herramientas. En el resto de esta sección se describe cómo configurar una región, comenzando por el enfoque más común.

Configuración de la región mediante el archivo compartido **`config`**

Defina la región configurando la `region` variable en el AWS `config` archivo compartido. Para obtener más información sobre el `config` archivo compartido, consulte los archivos de configuración y credenciales compartidos en la Guía de referencia de herramientas AWS SDKs y herramientas.

Ejemplo de configuración de este valor en el archivo `config`:

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

El archivo compartido `config` no se comprueba si la variable de entorno `AWS_SDK_CONFIG_OPT_OUT` está configurada.

Configuración de la región mediante variables de entorno

Configure la región mediante estableciendo la variable de entorno `AWS_REGION`.

Use el comando `export` para establecer esta variable en sistemas basados en Unix, como Linux o macOS. En el siguiente ejemplo se establece la región en `us-west-2`.

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

Para establecer esta variable en Windows, utilice el comando `set`. En el siguiente ejemplo se establece la región en `us-west-2`.

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

Configuración de la región con **`Aws.config`**

Configure la región añadiendo un valor `region` en el hash `Aws.config`. En el siguiente ejemplo se actualiza el hash `Aws.config` para utilizar la región `us-west-1`.

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

Todos los clientes o recursos que cree posteriormente estarán asociados a esta región.

Configuración de la región en un objeto de recurso

Establezca la región al crear un AWS cliente o un recurso. En el siguiente ejemplo se crea un objeto de recurso de Amazon S3; en la región `us-west-1`. Elija la región correcta para sus AWS recursos. Un objeto de cliente de servicio es inmutable, por lo que debe crear un cliente nuevo para cada servicio al que realice solicitudes y para realizar solicitudes al mismo servicio con una configuración diferente.

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

Configuración de un punto de conexión no estándar

La región se utiliza para crear un SSL punto final que se utilizará en AWS las solicitudes. Si necesita utilizar un punto de conexión no estándar en la región que ha seleccionado, añada una entrada de `endpoint` para `Aws.config`. Como alternativa, configure el `endpoint:` al crear un cliente de servicio o un objeto de recurso. En el siguiente ejemplo se crea un objeto de recurso de Amazon S3 en el punto de conexión `other_endpoint`.

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

Para usar el punto final que elija para API las solicitudes y mantener esa opción, consulte la opción de configuración de puntos finales específicos del servicio en la Guía de referencia de herramientas AWS SDKs y herramientas.

Usa el AWS SDK para Ruby

Esta sección proporciona información sobre el desarrollo de software AWS SDK para Ruby, incluida la forma de utilizar algunas de las funciones avanzadas SDK de Ruby.

La guía de referencia de herramientas AWS SDKs y herramientas también contiene configuraciones, características y otros conceptos fundamentales comunes a muchos de AWS SDKs ellos.

Temas

Usar la utilidad REPL de AWS SDK para Ruby

La gema `aws-sdk` incorpora una interfaz de línea de comandos interactiva Read-Eval-Print-Loop (REPL) en la que puede probar SDK para Ruby y ver los resultados de inmediato. Las gemas de SDK para Ruby están disponibles en RubyGems.org.

Requisitos previos

- Instalar AWS SDK para Ruby.
- El `aws-v3.rb` está ubicado en la gema `aws-sdk-resources`. La gema `aws-sdk-resources` también está incluida en la gema `aws-sdk` principal.
- Necesitará una biblioteca xml, como la gema `rexml`.
- Si bien el programa funciona con Interactive Ruby Shell (`irb`), recomendamos instalar la gema `pry` porque proporciona un entorno REPL más potente.

Configuración de Bundler

Si utiliza Bundler, las siguientes actualizaciones de su archivo `Gemfile` abordarán las gemas necesarias:

1. Abra el archivo `Gemfile` que creó al instalar el AWS SDK para Ruby. Añada las líneas siguientes al archivo:

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

2. Guarde el archivo Gemfile.
3. Instale las dependencias especificadas en su archivo `Gemfile`:

```
$ bundle install
```

Ejecución de REPL

Puede obtener acceso a REPL ejecutando `aws-v3.rb` en la línea de comandos.

```
aws-v3.rb
```

Si lo prefiere, puede habilitar el registro de red HTTP mediante la configuración del indicador detallado. El registro de red HTTP proporciona información sobre la comunicación entre el AWS SDK para Ruby y AWS. Tenga en cuenta que el indicador detallado también añade una sobrecarga que puede enlentecer la ejecución del código.

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

SDK para Ruby incluye clases de cliente que proporcionan interfaces para los Servicios de AWS. Cada clase de cliente admite un Servicio de AWS determinado. En la utilidad REPL, cada clase de servicio tiene un auxiliar que devuelve un nuevo objeto de cliente para interactuar con ese servicio. El nombre del auxiliar será el nombre del servicio convertido a minúsculas. Por ejemplo, los nombres de los objetos auxiliares de Amazon S3 y Amazon EC2 son `s3` y `ec2`, respectivamente. Para enumerar los buckets de Amazon S3 de su cuenta, puede introducir `s3.list_buckets` en el cuadro de diálogo.

Puede escribir `quit` en el mensaje de REPL para salir.

SDKUtilízalo con Ruby on Rails

Ruby on Rails proporciona un marco de desarrollo web que facilita la creación de sitios web con Ruby.

AWS proporciona la `aws-sdk-rails` gema que permite una fácil integración con Rails. Puede usar AWS Elastic Beanstalk AWS OpsWorks AWS CodeDeploy, o el AWS Rails Provisioner para implementar y ejecutar sus aplicaciones de Rails en la AWS nube.

Para obtener información sobre la instalación y el uso de la `aws-sdk-rails` gema, consulta el GitHub repositorio https://github.com/aws/aws-sdk-rails.

Consejo de depuración: Obtener información del rastreo de red de un cliente

Puede obtener información del rastro de red de un cliente de AWS configurando el `http_wire_trace` booleano. La información del rastro de red ayuda a diferenciar los cambios de cliente, los problemas de servicio y los errores de los usuarios. Cuando se define en `true`, la configuración muestra lo que se envía en la red. En el siguiente ejemplo se crea un cliente de Amazon S3 con rastro de red habilitado en el momento de la creación del cliente.

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

Dado el código y el argumento `bucket_name` siguientes, la salida muestra un mensaje que indica si existe un bucket con ese nombre.

```
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
```

Si el bucket existe, el resultado es similar al siguiente. (Las devoluciones se añaden a la línea HEAD para favorecer la legibilidad).

```
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
```

También puede activar el rastreo de red después de crear el cliente.

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

Para obtener más información sobre los campos de la información del rastreo de red, consulte los encabezados de solicitud obligatorios de Transfer Family.

Disociar respuestas y errores de cliente

Obtenga información sobre cómo disociar respuestas de cliente y errores de cliente en una aplicación AWS SDK para Ruby.

Disociación de respuestas de cliente

Cuando disocia una respuesta, AWS SDK para Ruby deshabilita el tráfico de red y el cliente devuelve datos disociados (o falsos). Si no se suministran datos de disociados, el cliente devuelve:

- Las listas como matrices vacías
- Los mapas como hashes vacíos
- Los valores numéricos como cero
- Las fechas como now

El siguiente ejemplo devuelve nombres disociados para la lista de buckets de Amazon S3.

```
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
```

La ejecución de este código muestra lo siguiente.

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

> **Note**
>
> Una vez proporcionados datos disociados, se dejan de aplicar los valores predeterminados a los atributos de instancia restantes. Esto significa que, en el ejemplo anterior, el atributo de instancia restante `creation_date` no es now sino `nil`.

AWS SDK para Ruby valida sus datos disociados. Si pasa datos del tipo equivocado, se genera una excepción `ArgumentError`. Por ejemplo, si en lugar de la asignación anterior a `bucket_data`, se hubiera usado lo siguiente:

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

AWS SDK para Ruby generara dos excepciones `ArgumentError`.

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

Disociación de errores de cliente

También puede proporcionar datos disociados para los errores que AWS SDK para Ruby genera para métodos específicos. El ejemplo siguiente muestra `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
```

Paginación

Algunas llamadas a AWS proporcionan respuestas paginadas para limitar la cantidad de datos que se devuelve con cada respuesta. Una página de datos representa un máximo de 1000 elementos.

Las respuestas paginadas se pueden enumerar

La forma más sencilla de administrar datos de respuesta paginados es usar el enumerador integrado en el objeto de respuesta, como se muestra en el siguiente ejemplo.

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

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

Esto produce un objeto de respuesta por cada llamada realizada a la API y enumera los objetos del bucket designado. El SDK recupera páginas adicionales de datos para completar la solicitud.

Administrar respuestas paginadas manualmente

Si desea controlar la paginación por sí mismo, use el método `next_page?` de la respuesta para comprobar si hay más páginas que pueden recuperarse o use el método `last_page?` para verificar que no hay más páginas que puedan recuperarse.

Si hay más páginas, utilice el método `next_page` (observe que no es ?) para recuperar la siguiente página de resultados, como se muestra en el siguiente ejemplo.

```
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**
>
> Si llama al método `next_page` y no hay más páginas que puedan recuperarse, el SDK genera una excepción Aws::PageableResponse::LastPageError.

Clases de datos paginados

Los datos paginados en AWS SDK para Ruby se administran a través de la clase Aws::PageableResponse, que se incluye con Seahorse::Client::Response para proporcionar acceso a los datos paginados.

Esperadores

Los esperadores son métodos de utilidad que sondean si un determinado estado se produce en un cliente. Los esperadores pueden fallar transcurrido un número de intentos en un intervalo de sondeo definido para el cliente del servicio. Para ver un ejemplo de cómo se usa un esperador, consulte el método create_table de Amazon DynamoDB Encryption Client en el repositorio de ejemplos de código AWS.

Invocación de un esperador

Para invocar un esperador, llame a `wait_until` en el cliente del servicio. En el siguiente ejemplo, un esperador espera hasta que la instancia `i-12345678` se esté ejecutando antes de continuar.

```
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
```

El primer parámetro corresponde al nombre del esperador, que es específico del cliente del servicio, e indica la operación que se está esperado. El segundo parámetro es un hash de los parámetros que se transfieren al método del cliente que ha llamado el esperador, lo cual varía según el nombre del esperador.

Para obtener una lista de las operaciones a las que el esperador puede esperar y los métodos del cliente que se llaman en cada operación, consulte la documentación de los campos `waiter_names` y `wait_until` para el cliente que esté utilizando.

Errores de espera

Los esperadores pueden fallar con cualquiera de las siguientes excepciones.

Aws::Waiters::Errors::FailureStateError

Se ha producido un estado de error durante la espera.

Aws::Waiters::Errors::NoSuchWaiterError

No se ha definido el nombre del esperador especificado para el cliente que se está utilizando.

Aws::Waiters::Errors::TooManyAttemptsError

El número de intentos ha superado el valor `max_attempts` del esperador.

Aws::Waiters::Errors::UnexpectedError

Se ha producido un error inesperado durante la espera.

Aws::Waiters::Errors::WaiterFailed

Se ha superado uno de los estados de espera o se ha producido otro error durante la espera.

Todos estos errores, excepto `NoSuchWaiterError`, se basan en `WaiterFailed`. Para detectar errores en un esperador, utilice `WaiterFailed`, tal y como se muestra en el siguiente ejemplo.

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

Configuración de un esperador

Cada esperador tiene un intervalo de sondeo predeterminado y un número máximo de intentos que hará antes de devolver el control al programa. Para establecer estos valores, utilice los parámetros `max_attempts` y `delay:` en la llamada a `wait_until`. En el siguiente ejemplo se espera hasta 25 segundos, realizando un sondeo cada cinco segundos.

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

Para deshabilitar los errores de espera, establezca el valor de cualquiera de estos parámetros en `nil`.

Ampliación de un esperador

Para modificar el comportamiento de los esperadores, puede registrar las devoluciones que se activan antes de cada intento de sondeo y antes del periodo de espera.

En el siguiente ejemplo se implementa un retardo exponencial en un esperador duplicando el tiempo que se debe esperar en cada intento.

```
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
```

En el siguiente ejemplo se deshabilita el número máximo de intentos y, en su lugar, se espera una hora (3.600 segundos) antes de que se produzca el error.

```
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
```

Especificar el comportamiento de reintento del cliente

De forma predeterminada, AWS SDK para Ruby realiza hasta tres reintentos, dejando transcurrir 15 segundos entre los reintentos, lo que equivale a un total de cuatro intentos. Por lo tanto, una operación podría tardar hasta 60 segundos en agotar el tiempo de espera.

El siguiente ejemplo crea un cliente de Amazon S3 en la región `us-west-2` y especifica esperar cinco segundos entre dos reintentos en cada operación de cliente. Por lo tanto, las operaciones de cliente de Amazon S3 podrían tardar hasta 15 segundos en agotar el tiempo de espera.

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

En este ejemplo se muestra cómo cambiar los parámetros de reintento directamente en el código. Sin embargo, también puede usar variables de entorno o el archivo compartido `config` de AWS para configurarlas para su aplicación. Para obtener más información sobre estos ajustes, consulte Comportamiento de reintento en la Guía de referencia de las herramientas y los SDK de AWS. Cualquier configuración explícita establecida en el código o en el propio cliente de servicio tiene prioridad sobre la establecida en las variables de entorno o en el archivo compartido `config`.

Migrar de la versión 1 o 2 a la versión 3 de AWS SDK para Ruby

El objetivo de este tema es ayudar a migrar de la versión 1 o 2 de AWS SDK para Ruby a la versión 3.

Uso en paralelo

No es necesario reemplazar la versión 1 o 2 de AWS SDK para Ruby por la versión 3. Puede utilizarlas de forma conjunta en la misma aplicación. Para obtener más información, consulte esta entrada del blog.

Un ejemplo rápido.

```
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
```

No es necesario volver a escribir el código funcional de la versión 1 o 2 existente para comenzar a utilizar el SDK de la versión 3. Una estrategia de migración válida es escribir solo el nuevo código con respecto al SDK de la versión 3.

Diferencias generales

La versión 3 difiere de la versión 2 en un aspecto importante.

- Cada servicio está disponible como una gema independiente.

La versión 2 difiere de la versión 1 en varios aspectos importantes.

- El espacio de nombres raíz es distinto: `Aws` frente a `AWS`. Esto permite realizar un uso en paralelo.

- `Aws.config`: ahora es un hash de Ruby simple, en lugar de un método.
- Estrictas opciones del constructor: al construir un cliente o un objeto de recurso en el SDK de la versión 1, no se tienen en cuenta las opciones del constructor desconocidas. En la versión 2, las opciones del constructor desconocidas activan un `ArgumentError`. Por ejemplo:

```
# 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
```

Diferencias del cliente

No existen importantes diferencias entre las clases de cliente de la versión 2 y la versión 3.

Entre la versión 1 y 2, las clases del cliente presentan menos diferencias externas. Muchos clientes de servicios tendrán interfaces compatibles tras la construcción del cliente. Algunas de las diferencias más destacadas:

- `Aws::S3::Client`: en la versión 1, la clase del cliente de Amazon S3 tiene codificación manual. La versión 2 se genera a partir de un modelo de servicio. Los nombres de método y las entradas son muy diferentes en la versión 2.
- `Aws::EC2::Client`: la versión 2 utiliza nombre en plural para las listas de salida, mientras que la versión 1 utiliza el sufijo `_set`. Por ejemplo:

```
# 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`: la versión 2 utiliza respuestas estructuradas, mientras que la versión 1 utiliza hash de Ruby simples.

- Nombres distintos según la clase de servicio: la versión 2 utiliza un nombre diferente para los distintos servicios:
  - `AWS::SimpleWorkflow` se ha convertido en `Aws::SWF`
  - `AWS::ELB` se ha convertido en `Aws::ElasticLoadBalancing`
  - `AWS::SimpleEmailService` se ha convertido en `Aws::SES`
- Opciones de configuración de clientes: algunas de las opciones de configuración de la versión 1 han cambiado de nombre en la versión 2. Otras se han eliminado o reemplazado. A continuación, se muestran los principales cambios:
  - Se ha eliminado `:use_ssl`. La versión 2 utiliza SSL para todo. Para deshabilitar SSL debe configurar un `:endpoint` que utilice `http://`.
  - `:ssl_ca_file` ahora es `:ssl_ca_bundle`
  - `:ssl_ca_path` ahora es `:ssl_ca_directory`
  - `:ssl_ca_store` añadido.
  - Ahora `:endpoint` debe ser un URI HTTP o HTTPS completo en lugar de un nombre de host.
  - Se han quitado las opciones `:*_port` de cada servicio y ahora se han reemplazado por `:endpoint`.
  - `:user_agent_prefix` ahora es `:user_agent_suffix`

Diferencias en los recursos

No existen importantes diferencias entre las interfaces de recurso de la versión 2 y la versión 3.

Existen importantes diferencias entre las interfaces de recurso de la versión 1 y la versión 2. Toda la versión 1 está codificada manualmente, mientras que las interfaces de recurso de la versión 2 se generan a partir de un modelo. Las interfaces de recurso de la versión 2 son significativamente más coherentes. Algunas de las diferencias entre sistemas son:

- Clase de recurso independiente: en la versión 2, el nombre del servicio es un módulo, no una clase. En este módulo se encuentra la interfaz de recurso:

```
# version 1
s3 = AWS::S3.new

# version 2
s3 = Aws::S3::Resource.new
```

- Referencia a los recursos: el SDK de la versión 2 separa los métodos getter de las colecciones y los recursos individuales en dos métodos diferentes:

```
# version 1
s3.buckets['bucket-name'].objects['key'].delete

# version 2
s3.bucket('bucket-name').object('key').delete
```

- Operaciones por lotes: en la versión 1, todas las operaciones por lotes eran utilidades con codificación manual. En la versión 2, muchas operaciones por lotes son operaciones procesadas por lotes de generación automática a través de la API. Las interfaces procesadas por lotes de la versión 2 son muy distintas de la versión 1.

SDKpara ejemplos de código de Ruby

En los ejemplos de código de este tema se muestra cómo utilizar el AWS SDK for Ruby with AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Servicios

- Ejemplos de Aurora usando SDK Ruby
- Ejemplos de Auto Scaling usando SDK Ruby
- CloudTrail ejemplos que utilizan SDK para Ruby
- CloudWatch ejemplos que utilizan SDK para Ruby
- Ejemplos de proveedores de identidad de Amazon Cognito que utilizan Ruby SDK
- Amazon Comprehend: ejemplos de uso de Ruby SDK
- Ejemplos de Amazon DocumentDB que utilizan Ruby SDK
- Ejemplos de DynamoDB usando Ruby SDK
- EC2Ejemplos de Amazon que utilizan SDK Ruby
- Ejemplos de Elastic SDK Beanstalk usando Ruby
- EventBridge ejemplos que utilizan SDK para Ruby
- AWS Glue ejemplos que utilizan SDK para Ruby
- IAMejemplos que utilizan SDK para Ruby
- Ejemplos de uso SDK de Kinesis para Ruby
- AWS KMS ejemplos que utilizan SDK Ruby
- Ejemplos de Lambda que utilizan Ruby SDK
- MSKEjemplos de Amazon que utilizan SDK Ruby
- Ejemplos de Amazon Polly que utilizan Ruby SDK
- RDSEjemplos de Amazon que utilizan SDK Ruby

- Ejemplos de Amazon S3 que utilizan SDK Ruby
- SESEjemplos de Amazon que utilizan SDK Ruby
- Ejemplos de Amazon SES API v2 que utilizan SDK Ruby
- SNSEjemplos de Amazon que utilizan SDK Ruby
- SQSEjemplos de Amazon que utilizan SDK Ruby
- AWS STS ejemplos que utilizan SDK para Ruby
- Ejemplos de Amazon Textract usando Ruby SDK
- Ejemplos de Amazon Translate que utilizan SDK Ruby
- WorkDocs Ejemplos de Amazon que utilizan SDK Ruby

Ejemplos de Aurora usando SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS SDK for Ruby de Aurora.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Aurora

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Aurora.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte D escribeDBClusters en la AWS SDK for Ruby APIreferencia.

Ejemplos de Auto Scaling usando SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Ruby uso de Auto Scaling.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción al escalado automático

En los siguientes ejemplos de código se muestra cómo empezar a utilizar el escalado automático.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte DescribeAutoScalingGroupsla AWS SDK for Ruby APIReferencia.

CloudTrail ejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with CloudTrail.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`CreateTrail`

En el siguiente ejemplo de código, se muestra cómo usar `CreateTrail`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte CreateTrailla AWS SDK for Ruby APIReferencia.

`DeleteTrail`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteTrail`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para API obtener más información, consulte DeleteTrailla AWS SDK for Ruby APIReferencia.

`ListTrails`

En el siguiente ejemplo de código, se muestra cómo usar `ListTrails`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListTrailsla AWS SDK for Ruby APIReferencia.

`LookupEvents`

En el siguiente ejemplo de código, se muestra cómo usar `LookupEvents`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte LookupEventsla AWS SDK for Ruby APIReferencia.

CloudWatch ejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with CloudWatch.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`DescribeAlarms`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeAlarms`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte DescribeAlarmsla AWS SDK for Ruby APIReferencia.

`DescribeAlarmsForMetric`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeAlarmsForMetric`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
#
# @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.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      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__
```

- Para API obtener más información, consulte DescribeAlarmsForMetricla AWS SDK for Ruby APIReferencia.

`DisableAlarmActions`

En el siguiente ejemplo de código, se muestra cómo usar `DisableAlarmActions`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return 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: "doc-example-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__
```

- Para API obtener más información, consulte DisableAlarmActionsla AWS SDK for Ruby APIReferencia.

`ListMetrics`

En el siguiente ejemplo de código, se muestra cómo usar `ListMetrics`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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__
```

- Para API obtener más información, consulte ListMetricsla AWS SDK for Ruby APIReferencia.

`PutMetricAlarm`

En el siguiente ejemplo de código, se muestra cómo usar `PutMetricAlarm`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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: 'doc-example-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
  )
  return true
```

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

- Para API obtener más información, consulte PutMetricAlarmla AWS SDK for Ruby APIReferencia.

`PutMetricData`

En el siguiente ejemplo de código, se muestra cómo usar `PutMetricData`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- Para API obtener más información, consulte PutMetricDatala AWS SDK for Ruby APIReferencia.

Ejemplos de proveedores de identidad de Amazon Cognito que utilizan Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el AWS SDK for Ruby uso de Amazon Cognito Identity Provider.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon Cognito

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon Cognito.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListUserPoolsla AWS SDK for Ruby APIReferencia.

Amazon Comprehend: ejemplos de uso de Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby con Amazon Comprehend.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Escenarios

Escenarios

Creación de una aplicación para analizar los comentarios de los clientes

El siguiente ejemplo de código muestra cómo crear una aplicación que analice las tarjetas de comentarios de los clientes, las traduzca del idioma original, determine sus opiniones y genere un archivo de audio a partir del texto traducido.

SDKpara Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para obtener el código fuente y las instrucciones de implementación, consulte el proyecto en GitHub.

Servicios utilizados en este ejemplo

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

Ejemplos de Amazon DocumentDB que utilizan Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Amazon DocumentDB. AWS SDK for Ruby

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Ejemplos sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon DocumentDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento que se desencadena al recibir registros de un flujo de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de Amazon DocumentDB con Lambda mediante Ruby.

```
require 'json'

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

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

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

```
end
```

Ejemplos de DynamoDB usando Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS SDK for Ruby

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a DynamoDB

En los siguientes ejemplos de código, se muestra cómo empezar a utilizar DynamoDB.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListTablesla AWS SDK for Ruby APIReferencia.

Temas

- Conceptos básicos
- Acciones

- Escenarios
- Ejemplos sin servidor

Conceptos básicos

Aprenda los conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de una tabla que pueda contener datos de películas.
- Colocar, obtener y actualizar una sola película en la tabla.
- Escriba los datos de la película en la tabla a partir de un JSON archivo de muestra.
- Consultar películas que se hayan estrenado en un año determinado.
- Buscar películas que se hayan estrenado en un intervalo de años.
- Eliminación de una película de la tabla y, a continuación, eliminar la tabla.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crear una clase que encapsula una tabla de DynamoDB.

```
# 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"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @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
```

Cree una función auxiliar para descargar y extraer el JSON archivo de muestra.

```
  # 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
```

Ejecutar un escenario interactivo para crear la tabla y realizar acciones en ella.

```
 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 if !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
```

- Para API obtener más información, consulte los siguientes temas en AWS SDK for Ruby APIReference.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

Acciones

`BatchExecuteStatement`

En el siguiente ejemplo de código, se muestra cómo usar `BatchExecuteStatement`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Leer un lote de elementos con PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :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
```

Eliminar un lote de elementos con PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte BatchExecuteStatementla AWS SDK for Ruby APIReferencia.

`BatchWriteItem`

En el siguiente ejemplo de código, se muestra cómo usar `BatchWriteItem`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte BatchWriteItemla AWS SDK for Ruby APIReferencia.

`CreateTable`

En el siguiente ejemplo de código, se muestra cómo usar `CreateTable`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource
  attr_reader :table_name
  attr_reader :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"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @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
```

- Para API obtener más información, consulte CreateTablela AWS SDK for Ruby APIReferencia.

`DeleteItem`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteItem`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte DeleteItemla AWS SDK for Ruby APIReferencia.

`DeleteTable`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteTable`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para API obtener más información, consulte DeleteTablela AWS SDK for Ruby APIReferencia.

`DescribeTable`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeTable`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

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

- Para API obtener más información, consulte DescribeTablela AWS SDK for Ruby APIReferencia.

`ExecuteStatement`

En el siguiente ejemplo de código, se muestra cómo usar `ExecuteStatement`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Seleccionar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :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
```

Actualizar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :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
```

Añadir un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :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
```

Eliminar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte ExecuteStatementla AWS SDK for Ruby APIReferencia.

`GetItem`

En el siguiente ejemplo de código, se muestra cómo usar `GetItem`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte GetItemla AWS SDK for Ruby APIReferencia.

`ListTables`

En el siguiente ejemplo de código, se muestra cómo usar `ListTables`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Determinar si existe una tabla.

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

- Para API obtener más información, consulte ListTablesla AWS SDK for Ruby APIReferencia.

`PutItem`

En el siguiente ejemplo de código, se muestra cómo usar `PutItem`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte PutItemla AWS SDK for Ruby APIReferencia.

`Query`

En el siguiente ejemplo de código, se muestra cómo usar `Query`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte Query in AWS SDK for Ruby APIReference.

`Scan`

En el siguiente ejemplo de código, se muestra cómo usar `Scan`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte Escanear en AWS SDK for Ruby APIreferencia.

`UpdateItem`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateItem`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :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
```

- Para API obtener más información, consulte UpdateItemla AWS SDK for Ruby APIReferencia.

Escenarios

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga un lote de elementos ejecutando varias SELECT declaraciones.
- Agregue un lote de elementos mediante la ejecución de varias INSERT declaraciones.
- Actualice un lote de elementos mediante la ejecución de varias UPDATE declaraciones.
- Elimine un lote de elementos mediante la ejecución de varias DELETE declaraciones.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario que crea una tabla y ejecuta lotes de consultas PartiQL.

```
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.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
  end
end
```

- Para API obtener más información, consulte BatchExecuteStatementla AWS SDK for Ruby APIReferencia.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga un elemento mediante la ejecución de una SELECT declaración.
- Agregue un elemento mediante la ejecución de una INSERT declaración.
- Actualice un elemento mediante la ejecución de una UPDATE declaración.
- Elimine un elemento mediante la ejecución de una DELETE declaración.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario que crea una tabla y ejecuta consultas PartiQL.

```
  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}".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.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
  end
end
```

- Para API obtener más información, consulte ExecuteStatementla AWS SDK for Ruby APIReferencia.

Ejemplos sin servidor

Invocación de una función de Lambda desde un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una transmisión de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de DynamoDB con Lambda mediante Ruby.

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

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

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

  def log_dynamodb_record(record)
    puts record['eventID']
```

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

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una transmisión de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Ruby.

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

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

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

EC2Ejemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonEC2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hola Amazon EC2

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonEC2.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte DescribeSecurityGroupsla AWS SDK for Ruby APIReferencia.

Temas

- Acciones

Acciones

`AllocateAddress`

En el siguiente ejemplo de código, se muestra cómo usar `AllocateAddress`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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")
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return "Error"
```

```
end
```

- Para API obtener más información, consulte AllocateAddressla AWS SDK for Ruby APIReferencia.

`AssociateAddress`

En el siguiente ejemplo de código, se muestra cómo usar `AssociateAddress`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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,
  )
  return response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return "Error"
end
```

- Para API obtener más información, consulte AssociateAddressla AWS SDK for Ruby APIReferencia.

`CreateKeyPair`

En el siguiente ejemplo de código, se muestra cómo usar `CreateKeyPair`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    "already exists."
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  return 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)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return 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__
```

- Para API obtener más información, consulte CreateKeyPairla AWS SDK for Ruby APIReferencia.

`CreateRouteTable`

En el siguiente ejemplo de código, se muestra cómo usar `CreateRouteTable`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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}'."
  return 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."
  return 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__
```

- Para API obtener más información, consulte CreateRouteTablela AWS SDK for Ruby APIReferencia.

`CreateSecurityGroup`

En el siguiente ejemplo de código, se muestra cómo usar `CreateSecurityGroup`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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}'."
  return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return "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}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

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

```
#
# - A security group with inbound rules, outbound rules, or both.
#
# @param p [Aws::EC2::Types::IpPermission] The IP permissions set.
# @example
#   ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
#   response = ec2_client.describe_security_groups
#   unless sg.ip_permissions.empty?
#     describe_security_group_permissions(
#       response.security_groups[0].ip_permissions[0]
#     )
#   end
def describe_security_group_permissions(perm)
  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

  unless perm.from_port.nil?
    if perm.from_port == "-1" || perm.from_port == -1
      print ", From: All"
    else
      print ", From: #{perm.from_port}"
    end
  end

  unless perm.to_port.nil?
    if perm.to_port == "-1" || perm.to_port == -1
      print ", To: All"
    else
      print ", To: #{perm.to_port}"
    end
  end

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

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

  print "\n"
end

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

```
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |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}"

      if sg.tags.count.positive?
        puts "Tags:"
        sg.tags.each do |tag|
          puts "  Key: #{tag.key}, Value: #{tag.value}"
        end
      end

      unless sg.ip_permissions.empty?
        puts "Inbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions.each do |p|
          describe_security_group_permissions(p)
        end
      end

      unless sg.ip_permissions_egress.empty?
        puts "Outbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions_egress.each do |p|
          describe_security_group_permissions(p)
        end
      end
    end
  else
    puts "No security groups found."
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end
```

```
# Deletes 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 to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )
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}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Example usage:
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 = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    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"
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  group_name = "my-security-group"
  description = "This is my security group."
  vpc_id = "vpc-6713dfEX"
  ip_protocol_http = "tcp"
  from_port_http = "80"
  to_port_http = "80"
  cidr_ip_range_http = "0.0.0.0/0"
  ip_protocol_ssh = "tcp"
  from_port_ssh = "22"
  to_port_ssh = "22"
  cidr_ip_range_ssh = "0.0.0.0/0"
  # 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
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

security_group_id = ""
security_group_exists = false
ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to create security group..."
security_group_id = create_security_group(
  ec2_client,
  group_name,
```

```
  description,
  vpc_id
)
if security_group_id == "Error"
  puts "Could not create security group. Skipping this step."
else
  security_group_exists = true
end

if security_group_exists
  puts "Attempting to add inbound rules to security group..."
  unless security_group_ingress_authorized?(
    ec2_client,
    security_group_id,
    ip_protocol_http,
    from_port_http,
    to_port_http,
    cidr_ip_range_http
  )
    puts "Could not add inbound HTTP rule to security group. " \
      "Skipping this step."
  end

  unless security_group_ingress_authorized?(
    ec2_client,
    security_group_id,
    ip_protocol_ssh,
    from_port_ssh,
    to_port_ssh,
    cidr_ip_range_ssh
  )
    puts "Could not add inbound SSH rule to security group. " \
      "Skipping this step."
  end
end

puts "\nInformation about available security groups:"
describe_security_groups(ec2_client)

if security_group_exists
  puts "\nAttempting to delete security group..."
  unless security_group_deleted?(ec2_client, security_group_id)
    puts "Could not delete security group. You must delete it yourself."
  end
```

```
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obtener más información, consulte CreateSecurityGroupla AWS SDK for Ruby APIReferencia.

`CreateSubnet`

En el siguiente ejemplo de código, se muestra cómo usar `CreateSubnet`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return 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__
```

- Para API obtener más información, consulte CreateSubnetla AWS SDK for Ruby APIReferencia.

`CreateVpc`

En el siguiente ejemplo de código, se muestra cómo usar `CreateVpc`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return 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__
```

- Para API obtener más información, consulte CreateVpcla AWS SDK for Ruby APIReferencia.

`DescribeInstances`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeInstances`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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__
```

- Para API obtener más información, consulte DescribeInstancesla AWS SDK for Ruby APIReferencia.

`DescribeRegions`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeRegions`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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__
```

- Para API obtener más información, consulte DescribeRegionsla AWS SDK for Ruby APIReferencia.

`ReleaseAddress`

En el siguiente ejemplo de código, se muestra cómo usar `ReleaseAddress`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end
```

- Para API obtener más información, consulte ReleaseAddressla AWS SDK for Ruby APIReferencia.

`StartInstances`

En el siguiente ejemplo de código, se muestra cómo usar `StartInstances`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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."
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return 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)..."
  unless instance_started?(ec2_client, instance_id)
    puts "Could not start instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obtener más información, consulte StartInstancesla AWS SDK for Ruby APIReferencia.

`StopInstances`

En el siguiente ejemplo de código, se muestra cómo usar `StopInstances`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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."
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return 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)..."
  unless instance_stopped?(ec2_client, instance_id)
    puts "Could not stop instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obtener más información, consulte StopInstancesla AWS SDK for Ruby APIReferencia.

`TerminateInstances`

En el siguiente ejemplo de código, se muestra cómo usar `TerminateInstances`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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."
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return 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)..."
  unless instance_terminated?(ec2_client, instance_id)
    puts "Could not terminate instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obtener más información, consulte TerminateInstancesla AWS SDK for Ruby APIReferencia.

Ejemplos de Elastic SDK Beanstalk usando Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby con Elastic Beanstalk.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`DescribeApplications`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeApplications`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DescribeApplicationsla AWS SDK for Ruby APIReferencia.

`ListAvailableSolutionStacks`

En el siguiente ejemplo de código, se muestra cómo usar `ListAvailableSolutionStacks`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListAvailableSolutionStacksla AWS SDK for Ruby APIReferencia.

`UpdateApplication`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateApplication`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte UpdateApplicationla AWS SDK for Ruby APIReferencia.

EventBridge ejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with EventBridge.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Escenarios

Crear y activar una regla

El siguiente ejemplo de código muestra cómo crear y activar una regla en Amazon EventBridge.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Llamar a las funciones en el orden correcto.

```
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"
```

Comprueba si el tema especificado de Amazon Simple Notification Service (AmazonSNS) existe entre los proporcionados para esta función.

```
# 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
  return false
end
```

Comprueba si el tema especificado existe entre los disponibles para la persona que llama en AmazonSNS.

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

Crea un tema en Amazon SNS y, a continuación, suscríbete a una dirección de correo electrónico para recibir notificaciones sobre ese tema.

```
# 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."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Compruebe si el rol especificado AWS Identity and Access Management (IAM) existe entre los proporcionados a esta función.

```
# 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
  return false
```

```
end
```

Compruebe si el rol especificado existe entre los disponibles para la persona que llama. IAM

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

Crea un rol enIAM.

```
# 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."
  return 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."
  return "Error"
end
```

Comprueba si la EventBridge regla especificada existe entre las que se proporcionan a esta función.

```
# 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
  return false
```

```
end
```

Comprueba si la regla especificada existe entre las disponibles para la persona que llama. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end
```

Crea una regla en EventBridge.

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

```
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        "aws.ec2"
      ],
      'detail-type': [
        "EC2 Instance State-change Notification"
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: "ENABLED",
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return 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."
```

```
  return false
end
```

Comprueba si el grupo de registros especificado existe entre los disponibles para la persona que llama en Amazon CloudWatch Logs.

```
# 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."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

Cree un grupo de CloudWatch registros en Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
```

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

Escribe un evento en una secuencia de CloudWatch registros en Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
```

```
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

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

Reinicie una instancia de Amazon Elastic Compute Cloud (AmazonEC2) y añada información sobre la actividad relacionada a un flujo de CloudWatch registros en Logs.

```
# 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
  )

  return 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
  )
  return false
end
```

Muestra información sobre la actividad de una regla en EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
```

```
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
```

```
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Muestra la información de registro de todos los flujos de registros de un grupo de CloudWatch registros.

```
# 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
```

Muestra un recordatorio a la persona que llama para que limpie manualmente AWS los recursos asociados que ya no necesite.

```
# 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

# Example usage:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
  # Properties for the IAM role.
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
  # Properties for the Amazon EventBridge rule.
  rule_name = "aws-doc-sdk-examples-ec2-state-change"
  rule_description = "Triggers when any available EC2 instance starts."
  instance_state = "running"
  target_id = "sns-topic"
  # Properties for the Amazon EC2 instance.
  instance_id = "i-033c48ef067af3dEX"
  # Properties for displaying the event rule's activity.
  start_time = Time.now - 600 # Go back over the past 10 minutes
                              # (10 minutes * 60 seconds = 600 seconds).
  end_time = Time.now
  period = 60 # Look back every 60 seconds over the past 10 minutes.
  # Properties for the Amazon CloudWatch Logs log group.
  log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
  # AWS service clients for this code example.
  region = "us-east-1"
  sts_client = Aws::STS::Client.new(region: region)
  sns_client = Aws::SNS::Client.new(region: region)
  iam_client = Aws::IAM::Client.new(region: region)
  cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
  ec2_client = Aws::EC2::Client.new(region: region)
  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

  # Get the caller's account ID for use in forming
  # Amazon Resource Names (ARNs) that this code relies on later.
  account_id = sts_client.get_caller_identity.account

  # If the Amazon SNS topic doesn't exist, create it.
```

```
topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
unless topic_exists?(sns_client, topic_arn)
  topic_arn = create_topic(sns_client, topic_name, email_address)
  if topic_arn == "Error"
    puts "Could not create the Amazon SNS topic correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
unless role_exists?(iam_client, role_arn)
  role_arn = create_role(iam_client, role_name)
  if role_arn == "Error"
    puts "Could not create the IAM role correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
  unless rule_created?(
    cloudwatchevents_client,
    rule_name,
    rule_description,
    instance_state,
    role_arn,
    target_id,
    topic_arn
  )
    puts "Could not create the Amazon EventBridge rule correctly. " \
      "Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
```

```
  unless log_group_exists?(cloudwatchlogs_client, log_group_name)
    unless log_group_created?(cloudwatchlogs_client, log_group_name)
      puts "Could not create the Amazon CloudWatch Logs log group " \
      "correctly. Program stopped."
      manual_cleanup_notice(
        topic_name, role_name, rule_name, log_group_name, instance_id
      )
    end
  end

  # Restart the Amazon EC2 instance, which triggers the rule.
  unless instance_restarted?(
    ec2_client,
    cloudwatchlogs_client,
    instance_id,
    log_group_name
  )
    puts "Could not restart the instance to trigger the rule. " \
      "Continuing anyway to show information about the rule and logs..."
  end

  # Display how many times the rule was triggered over the past 10 minutes.
  display_rule_activity(
    cloudwatch_client,
    rule_name,
    start_time,
    end_time,
    period
  )

  # Display related log data in Amazon CloudWatch Logs.
  display_log_data(cloudwatchlogs_client, log_group_name)

  # Reminder the caller to clean up any AWS resources that are used
  # by this code example and are no longer needed.
  manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
  )
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obtener más información, consulte los siguientes temas en AWS SDK for Ruby APIReference.
  - PutEvents
  - PutRule

AWS Glue ejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS Glue.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

¿Hola AWS Glue

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS Glue.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListJobsla AWS SDK for Ruby APIReferencia.

Temas

- Conceptos básicos
- Acciones

Conceptos básicos

Aprenda los conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un rastreador que rastree un bucket público de Amazon S3 y genere una base de datos de CSV metadatos con formato.
- Enumere información sobre bases de datos y tablas en su. AWS Glue Data Catalog
- Cree un trabajo para extraer CSV datos del depósito de S3, transformarlos y cargar la salida JSON con formato en otro depósito de S3.
- Incluir información sobre las ejecuciones de trabajos, ver algunos de los datos transformados y limpiar los recursos.

Para obtener más información, consulte el tutorial: Primeros pasos con AWS Glue Studio.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase que agrupe AWS Glue las funciones utilizadas en el escenario.

```
# 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
```

Crear una clase que ejecute el escenario.

```
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)

    new_step(1, "Create a crawler")
    puts "Checking for crawler #{crawler_name}."
    crawler = wrapper.get_crawler(crawler_name)
    if crawler == false
      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}:"
      crawler = wrapper.get_crawler(crawler_name)
      puts JSON.pretty_generate(crawler).yellow
    end
    print "\nDone!\n".green

    new_step(2, "Run a crawler to output a database.")
    puts "Location of input data analyzed by crawler: #{data_source}"
    puts "Outputs: a Data Catalog database in CSV format containing metadata on
 input."
    wrapper.start_crawler(crawler_name)
    puts "Starting crawler... (this typically takes a few minutes)"
    crawler_state = nil
    while crawler_state != "READY"
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]["state"]
      print "Status check: #{crawler_state}.".yellow
    end
    print "\nDone!\n".green

    new_step(3, "Query the database.")
    database = wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
```

```
    print "#{database}".yellow
    puts "\nThe database contains these tables:"
    tables = wrapper.get_tables(db_name)
    tables.each_with_index do |table, index|
      print "\t#{index + 1}. #{table['name']}".yellow
    end
    print "\nDone!\n".green

    new_step(4, "Create a job definition that runs an ETL script.")
    puts "Uploading Python ETL script to S3..."
    wrapper.upload_job_script(job_script, @glue_bucket)
    puts "Creating job definition #{job_name}:\n"
    response = wrapper.create_job(job_name, "Getting started example job.",
@glue_service_role.arn, "s3://#{@glue_bucket.name}/#{job_script}")
    puts JSON.pretty_generate(response).yellow
    print "\nDone!\n".green

    new_step(5, "Start a new job")
    job_run_status = nil
    job_run_id = wrapper.start_job_run(
      job_name,
      db_name,
      tables[0]["name"],
      @glue_bucket.name
    )
    puts "Job #{job_name} started. Let's wait for it to run."
    until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].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 "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
    end
    print "\nDone!\n".green

    new_step(6, "View results from a successful job run.")
    if job_run_status == "SUCCEEDED"
      puts "Data from your job run is stored in your S3 bucket
'#{@glue_bucket.name}'. Files include:"
      begin

        # Print the key name of each object in the bucket.
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            print "#{object_summary.key}".yellow
```

```
          end
        end

        # Print the first 256 bytes of a run file
        desired_sample_objects = 1
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            if desired_sample_objects > 0
              sample_object = @glue_bucket.object(object_summary.key)
              sample = sample_object.get(range: "bytes=0-255").body.read
              puts "\nSample run file contents:"
              print "#{sample}".yellow
              desired_sample_objects -= 1
            end
          end
        end
      rescue Aws::S3::Errors::ServiceError => e
        logger.error(
          "Couldn't get job run data. Here's why: %s: %s",
          e.response.error.code, e.response.error.message
        )
        raise
      end
    end
    print "\nDone!\n".green

    new_step(7, "Delete job definition and crawler.")
    wrapper.delete_job(job_name)
    puts "Job deleted: #{job_name}."
    wrapper.delete_crawler(crawler_name)
    puts "Crawler deleted: #{crawler_name}."
    wrapper.delete_table(db_name, tables[0]["name"])
    puts "Table deleted: #{tables[0]["name"]} in #{db_name}."
    wrapper.delete_database(db_name)
    puts "Database deleted: #{db_name}."
    print "\nDone!\n".green
  end
end

def main

  banner("../../helpers/banner.txt")
  puts
 "#################################################################################################
```

```
  puts "#
                          #".yellow
  puts "#                                                EXAMPLE CODE DEMO:
                          #".yellow
  puts "#                                                    AWS Glue
                          #".yellow
  puts "#
                          #".yellow
  puts
"#################################################################################################
  puts ""
  puts "You have launched a demo of AWS Glue using the AWS for Ruby v3 SDK. Over the
next 60 seconds, it will"
  puts "do the following:"
  puts "    1. Create a crawler."
  puts "    2. Run a crawler to output a database."
  puts "    3. Query the database."
  puts "    4. Create a job definition that runs an ETL script."
  puts "    5. Start a new job."
  puts "    6. View results from a successful job run."
  puts "    7. Delete job definition and crawler."
  puts ""

  confirm_begin
  billing
  security
  puts "\e[H\e[2J"

  # Set input file names
  job_script_filepath = "job_script.py"
  resource_names = YAML.load_file("resource_names.yaml")

  # Instantiate existing IAM role.
  iam = Aws::IAM::Resource.new(region: "us-east-1")
  iam_role_name = resource_names["glue_service_role"]
  iam_role = iam.role(iam_role_name)

  # Instantiate existing S3 bucket.
  s3 = Aws::S3::Resource.new(region: "us-east-1")
  s3_bucket_name = resource_names["glue_bucket"]
  s3_bucket = s3.bucket(s3_bucket_name)

  scenario = GlueCrawlerJobScenario.new(
    Aws::Glue::Client.new(region: "us-east-1"),
```

```
    iam_role,
    s3_bucket,
    @logger
  )

  random_int = rand(10 ** 4)
  scenario.run(
    "doc-example-crawler-#{random_int}",
    "doc-example-database-#{random_int}",
    "doc-example-#{random_int}-",
    "s3://crawler-public-us-east-1/flight/2016/csv",
    job_script_filepath,
    "doc-example-job-#{random_int}"
  )

  puts "-" * 88
  puts "You have reached the end of this tour of AWS Glue."
  puts "To destroy CDK-created resources, run:\n       cdk destroy"
  puts "-" * 88

end
```

Cree un ETL script que sirva AWS Glue para extraer, transformar y cargar datos durante la ejecución de los trabajos.

```
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()
```

- Para API obtener más información, consulte los siguientes temas en AWS SDK for Ruby APIReference.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Acciones

`CreateCrawler`

En el siguiente ejemplo de código, se muestra cómo usar `CreateCrawler`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte CreateCrawlerla AWS SDK for Ruby APIReferencia.

`CreateJob`

En el siguiente ejemplo de código, se muestra cómo usar `CreateJob`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte CreateJobla AWS SDK for Ruby APIReferencia.

`DeleteCrawler`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteCrawler`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteCrawlerla AWS SDK for Ruby APIReferencia.

`DeleteDatabase`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteDatabase`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteDatabasela AWS SDK for Ruby APIReferencia.

`DeleteJob`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteJob`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteJobla AWS SDK for Ruby APIReferencia.

`DeleteTable`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteTable`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteTablela AWS SDK for Ruby APIReferencia.

`GetCrawler`

En el siguiente ejemplo de código, se muestra cómo usar `GetCrawler`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetCrawlerla AWS SDK for Ruby APIReferencia.

`GetDatabase`

En el siguiente ejemplo de código, se muestra cómo usar `GetDatabase`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetDatabasela AWS SDK for Ruby APIReferencia.

`GetJobRun`

En el siguiente ejemplo de código, se muestra cómo usar `GetJobRun`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetJobRunla AWS SDK for Ruby APIReferencia.

`GetJobRuns`

En el siguiente ejemplo de código, se muestra cómo usar `GetJobRuns`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetJobRunsla AWS SDK for Ruby APIReferencia.

`GetTables`

En el siguiente ejemplo de código, se muestra cómo usar `GetTables`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetTablesla AWS SDK for Ruby APIReferencia.

`ListJobs`

En el siguiente ejemplo de código, se muestra cómo usar `ListJobs`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListJobsla AWS SDK for Ruby APIReferencia.

`StartCrawler`

En el siguiente ejemplo de código, se muestra cómo usar `StartCrawler`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte StartCrawlerla AWS SDK for Ruby APIReferencia.

`StartJobRun`

En el siguiente ejemplo de código, se muestra cómo usar `StartJobRun`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte StartJobRunla AWS SDK for Ruby APIReferencia.

IAMejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby withIAM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a IAM

Los siguientes ejemplos de código muestran cómo empezar a usarloIAM.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListPolicieslа AWS SDK for Ruby APIReferencia.

Temas

- Acciones
- Escenarios

Acciones

`AttachRolePolicy`

En el siguiente ejemplo de código, se muestra cómo usar `AttachRolePolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# 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
```

- Para API obtener más información, consulte AttachRolePolicyla AWS SDK for Ruby APIReferencia.

`AttachUserPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `AttachUserPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte AttachUserPolicyla AWS SDK for Ruby APIReferencia.

`CreateAccessKey`

En el siguiente ejemplo de código, se muestra cómo usar `CreateAccessKey`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# 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 => e
    @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 => e
    @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
```

- Para API obtener más información, consulte CreateAccessKeyla AWS SDK for Ruby APIReferencia.

`CreateAccountAlias`

En el siguiente ejemplo de código, se muestra cómo usar `CreateAccountAlias`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
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
```

- Para API obtener más información, consulte CreateAccountAliasla AWS SDK for Ruby APIReferencia.

`CreatePolicy`

En el siguiente ejemplo de código, se muestra cómo usar `CreatePolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# 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
```

- Para API obtener más información, consulte CreatePolicyla AWS SDK for Ruby APIReferencia.

`CreateRole`

En el siguiente ejemplo de código, se muestra cómo usar `CreateRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte CreateRolela AWS SDK for Ruby APIReferencia.

`CreateServiceLinkedRole`

En el siguiente ejemplo de código, se muestra cómo usar `CreateServiceLinkedRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte CreateServiceLinkedRolela AWS SDK for Ruby APIReferencia.

`CreateUser`

En el siguiente ejemplo de código, se muestra cómo usar `CreateUser`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte CreateUserla AWS SDK for Ruby APIReferencia.

`DeleteAccessKey`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteAccessKey`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# 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 => e
    @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 => e
    @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
```

- Para API obtener más información, consulte DeleteAccessKeyla AWS SDK for Ruby APIReferencia.

`DeleteAccountAlias`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteAccountAlias`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
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
```

- Para API obtener más información, consulte DeleteAccountAliasla AWS SDK for Ruby APIReferencia.

`DeleteRole`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    begin
      # Detach and delete attached policies
      @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
        response.attached_policies.each do |policy|
          @iam_client.detach_role_policy({
                                       role_name: role_name,
                                       policy_arn: policy.policy_arn
                                     })
          # Check if the policy is a customer managed policy (not AWS managed)
          unless policy.policy_arn.include?("aws:policy/")
            @iam_client.delete_policy({ policy_arn: policy.policy_arn })
            @logger.info("Deleted customer managed policy #{policy.policy_name}.")
          end
        end
      end

      # Delete the role
      @iam_client.delete_role({ role_name: role_name })
      @logger.info("Deleted role #{role_name}.")
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
 why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- Para API obtener más información, consulte DeleteRolela AWS SDK for Ruby APIReferencia.

`DeleteServerCertificate`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteServerCertificate`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
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
```

- Para API obtener más información, consulte DeleteServerCertificatela AWS SDK for Ruby APIReferencia.

`DeleteServiceLinkedRole`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteServiceLinkedRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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)
  unless e.code == "NoSuchEntity"
```

```
      @logger.error("Couldn't delete #{role_name}. Here's why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- Para API obtener más información, consulte DeleteServiceLinkedRolela AWS SDK for Ruby APIReferencia.

`DeleteUser`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteUser`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte DeleteUserla AWS SDK for Ruby APIReferencia.

`DeleteUserPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteUserPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte DeleteUserPolicyla AWS SDK for Ruby APIReferencia.

`DetachRolePolicy`

En el siguiente ejemplo de código, se muestra cómo usar `DetachRolePolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# 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
```

- Para API obtener más información, consulte DetachRolePolicyla AWS SDK for Ruby APIReferencia.

`DetachUserPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `DetachUserPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte DetachUserPolicyla AWS SDK for Ruby APIReferencia.

`GetAccountPasswordPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `GetAccountPasswordPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
    begin
      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
end
```

- Para API obtener más información, consulte GetAccountPasswordPolicyla AWS SDK for Ruby APIReferencia.

`GetPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `GetPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte GetPolicyla AWS SDK for Ruby APIReferencia.

`GetRole`

En el siguiente ejemplo de código, se muestra cómo usar `GetRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte GetRolela AWS SDK for Ruby APIReferencia.

`GetUser`

En el siguiente ejemplo de código, se muestra cómo usar `GetUser`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte GetUserla AWS SDK for Ruby APIReferencia.

`ListAccessKeys`

En el siguiente ejemplo de código, se muestra cómo usar `ListAccessKeys`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Este módulo de ejemplo muestra, crea, desactiva y elimina las claves de acceso.

```
# 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 => e
    @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 => e
    @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
```

- Para API obtener más información, consulte ListAccessKeysla AWS SDK for Ruby APIReferencia.

`ListAccountAliases`

En el siguiente ejemplo de código, se muestra cómo usar `ListAccountAliases`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, cree y elimine los alias de la cuenta.

```
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
```

- Para API obtener más información, consulte ListAccountAliasesla AWS SDK for Ruby APIReferencia.

`ListAttachedRolePolicies`

En el siguiente ejemplo de código, se muestra cómo usar `ListAttachedRolePolicies`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# 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
```

- Para API obtener más información, consulte ListAttachedRolePoliciesla AWS SDK for Ruby APIReferencia.

`ListGroups`

En el siguiente ejemplo de código, se muestra cómo usar `ListGroups`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListGroupsla AWS SDK for Ruby APIReferencia.

`ListPolicies`

En el siguiente ejemplo de código, se muestra cómo usar `ListPolicies`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

En este módulo de ejemplo, se enumeran, se crean, se adjuntan y se separan las políticas de roles.

```
# 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
```

- Para API obtener más información, consulte ListPolicieslа AWS SDK for Ruby APIReferencia.

`ListRolePolicies`

En el siguiente ejemplo de código, se muestra cómo usar `ListRolePolicies`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListRolePoliciesla AWS SDK for Ruby APIReferencia.

`ListRoles`

En el siguiente ejemplo de código, se muestra cómo usar `ListRoles`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListRoleslа AWS SDK for Ruby APIReferencia.

`ListSAMLProviders`

En el siguiente ejemplo de código, se muestra cómo usar `ListSAMLProviders`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte L istSAMLProviders en la AWS SDK for Ruby APIreferencia.

`ListServerCertificates`

En el siguiente ejemplo de código, se muestra cómo usar `ListServerCertificates`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
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
```

- Para API obtener más información, consulte ListServerCertificatesla AWS SDK for Ruby APIReferencia.

`ListUsers`

En el siguiente ejemplo de código, se muestra cómo usar `ListUsers`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte ListUsersla AWS SDK for Ruby APIReferencia.

`PutUserPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `PutUserPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte PutUserPolicyla AWS SDK for Ruby APIReferencia.

`UpdateServerCertificate`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateServerCertificate`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Enumere, actualice y elimine certificados del servidor.

```
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
```

- Para API obtener más información, consulte UpdateServerCertificatela AWS SDK for Ruby APIReferencia.

`UpdateUser`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateUser`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte UpdateUserla AWS SDK for Ruby APIReferencia.

Escenarios

Crear un usuario y asumir un rol

En el siguiente ejemplo de código, se muestra cómo crear un usuario y asumir un rol.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a IAM los usuarios para autenticarse cuando desarrolle software diseñado específicamente o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

- Crear un usuario que no tenga permisos.
- Crear un rol que conceda permiso para enumerar los buckets de Amazon S3 para la cuenta.

- Agregar una política para que el usuario asuma el rol.
- Asumir el rol y enumerar los buckets de S3 con credenciales temporales, y después limpiar los recursos.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree un IAM usuario y un rol que concedan permiso para publicar buckets de Amazon S3. El usuario solo tiene derechos para asumir el rol. Después de asumir el rol, use las credenciales temporales para enumerar los buckets de la cuenta.

```
# 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__
```

- Para API obtener más información, consulte los siguientes temas en la sección de AWS SDK for Ruby APIreferencia.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Ejemplos de uso SDK de Kinesis para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK for Ruby uso de Kinesis.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Ejemplos sin servidor

Ejemplos sin servidor

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de Kinesis con Lambda mediante Ruby.

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

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Ruby.

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

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

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

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

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

```
end
```

AWS KMS ejemplos que utilizan SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS KMS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`CreateKey`

En el siguiente ejemplo de código, se muestra cómo usar `CreateKey`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte CreateKeyla AWS SDK for Ruby APIReferencia.

`Decrypt`

En el siguiente ejemplo de código, se muestra cómo usar `Decrypt`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte Decrypt in AWS SDK for Ruby APIReference.

`Encrypt`

En el siguiente ejemplo de código, se muestra cómo usar `Encrypt`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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*")
```

- Para API obtener más información, consulte Cifrar en AWS SDK for Ruby APIreferencia.

`ReEncrypt`

En el siguiente ejemplo de código, se muestra cómo usar `ReEncrypt`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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*")
```

- Para API obtener más información, consulte ReEncryptla AWS SDK for Ruby APIReferencia.

Ejemplos de Lambda que utilizan Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Lambda.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Lambda

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Lambda.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListFunctionsla AWS SDK for Ruby APIReferencia.

Temas

- Acciones
- Escenarios
- Ejemplos sin servidor

Acciones

`CreateFunction`

En el siguiente ejemplo de código, se muestra cómo usar `CreateFunction`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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. This
  #                      must include the file name and the function name.
  # @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
```

- Para API obtener más información, consulte CreateFunctionla AWS SDK for Ruby APIReferencia.

`DeleteFunction`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteFunction`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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
```

- Para API obtener más información, consulte DeleteFunctionla AWS SDK for Ruby APIReferencia.

`GetFunction`

En el siguiente ejemplo de código, se muestra cómo usar `GetFunction`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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
```

- Para API obtener más información, consulte GetFunctionla AWS SDK for Ruby APIReferencia.

`Invoke`

En el siguiente ejemplo de código, se muestra cómo usar `Invoke`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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
```

- Para API obtener más información, consulte Invoke in AWS SDK for Ruby APIReference.

`ListFunctions`

En el siguiente ejemplo de código, se muestra cómo usar `ListFunctions`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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 executing #{function_name}:\n #{e.message}")
  end
```

- Para API obtener más información, consulte ListFunctionsla AWS SDK for Ruby APIReferencia.

`UpdateFunctionCode`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateFunctionCode`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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
```

- Para API obtener más información, consulte UpdateFunctionCodela AWS SDK for Ruby APIReferencia.

`UpdateFunctionConfiguration`

En el siguiente ejemplo de código, se muestra cómo usar `UpdateFunctionConfiguration`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @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
```

- Para API obtener más información, consulte UpdateFunctionConfigurationla AWS SDK for Ruby APIReferencia.

Escenarios

Creación de una aplicación para analizar los comentarios de los clientes

El siguiente ejemplo de código muestra cómo crear una aplicación que analice las tarjetas de comentarios de los clientes, las traduzca del idioma original, determine sus opiniones y genere un archivo de audio a partir del texto traducido.

SDKpara Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para obtener el código fuente y las instrucciones de implementación, consulte el proyecto en GitHub.

Servicios utilizados en este ejemplo

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

Comenzar a usar las funciones

En el siguiente ejemplo de código, se muestra cómo:

- Cree un IAM rol y una función Lambda y, a continuación, cargue el código del controlador.
- Invocar la función con un único parámetro y obtener resultados.
- Actualizar el código de la función y configurar con una variable de entorno.
- Invocar la función con un nuevo parámetro y obtener resultados. Mostrar el registro de ejecución devuelto.
- Enumerar las funciones de su cuenta y, luego, limpiar los recursos.

Para obtener información, consulte Crear una función de Lambda con la consola.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Configure los IAM permisos necesarios para una función Lambda capaz de escribir registros.

```
# 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)
  role_policy = {
```

```
      'Version': "2012-10-17",
      'Statement': [
        {
          'Effect': "Allow",
          'Principal': {
            'Service': "lambda.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }
    case action
    when "create"
      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
        }
      )
      $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
        w.max_attempts = 5
        w.delay = 5
      end
      @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
      @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
 fully.")
      sleep(10)
      return role, role_policy.to_json
    when "destroy"
      $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}")
    else
      raise "Incorrect action provided. Must provide 'create' or 'destroy'"
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating role or attaching policy:\n
 #{e.message}")
  end
```

Definir un controlador de Lambda que aumente un número dado como parámetro de invocación.

```
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
```

Comprimir su función de Lambda en un paquete de implementación.

```
 # Creates a Lambda deployment package in .zip format.
 # This zip can be passed directly as a string to Lambda when creating the
function.
 #
 # @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) {
     |zipfile|
     zipfile.add("lambda_function.rb", "#{source_file}.rb")
   }
   @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
```

Crear una nueva función de Lambda.

```
 # Deploys a Lambda function.
 #
 # @param function_name: The name of the Lambda function.
 # @param handler_name: The fully qualified name of the handler function. This
 #                      must include the file name and the function name.
 # @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
```

Invocar su función de Lambda con parámetros de tiempo de ejecución opcionales.

```
  # 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
```

Actualizar la configuración de su función de Lambda para introducir una nueva variable de entorno.

```
  # 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
```

Actualizar el código de su función de Lambda con un paquete de implementación diferente que contenga un código diferente.

```
  # 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
```

Enumerar todas las funciones de Lambda existentes mediante el paginador integrado.

```
  # 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 executing #{function_name}:\n #{e.message}")
  end
```

Eliminar una función de Lambda específica.

```
  # 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
```

- Para API obtener más información, consulte los siguientes temas en la sección de referencia.AWS SDK for Ruby API
  - CreateFunction

- DeleteFunction
- GetFunction
- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Ejemplos sin servidor

Conexión a una RDS base de datos de Amazon en una función Lambda

El siguiente ejemplo de código muestra cómo implementar una función Lambda que se conecta a una RDS base de datos. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Conexión a una RDS base de datos de Amazon en una función Lambda mediante Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
```

```
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Uso de un evento de Kinesis con Lambda mediante Ruby.

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

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Invocación de una función de Lambda desde un desencadenador de DynamoDB

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento que se desencadena al recibir registros de una transmisión de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de DynamoDB con Lambda mediante Ruby.

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

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

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

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

Invocación de una función de Lambda desde un desencadenador de Amazon DocumentDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento que se desencadena al recibir registros de un flujo de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de Amazon DocumentDB con Lambda mediante Ruby.

```
require 'json'

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

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

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

Invocar una función Lambda desde un disparador de Amazon MSK

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de registros de un clúster de AmazonMSK. La función recupera la MSK carga útil y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un MSK evento de Amazon con Lambda mediante Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

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

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a Amazon S3 API para recuperar y registrar el tipo de contenido del objeto.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de S3 con Lambda mediante Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Invocar una función Lambda desde un disparador de Amazon SNS

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de mensajes de un SNS tema. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un SNS evento con Lambda mediante Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Invocar una función Lambda desde un disparador de Amazon SQS

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de mensajes de una SQS cola. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un SQS evento con Lambda mediante Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Ruby.

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

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

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

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

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

```
end
```

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una transmisión de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Ruby.

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

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

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

Notificación de errores de artículos de lotes para funciones de Lambda con un activador de Amazon SQS

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una SQS cola. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Informe de errores de elementos de SQS lote con Lambda mediante Ruby.

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

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

MSKEjemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonMSK.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Ejemplos sin servidor

Invocar una función Lambda desde un disparador de Amazon MSK

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de registros de un clúster de AmazonMSK. La función recupera la MSK carga útil y registra el contenido del registro.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un MSK evento de Amazon con Lambda mediante Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
```

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

Ejemplos de Amazon Polly que utilizan Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon Polly.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Escenarios

Acciones

`DescribeVoices`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeVoices`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 => ex
  puts "Could not get voices"
  puts "Error message:"
  puts ex.message
end
```

- Para API obtener más información, consulte DescribeVoicesla AWS SDK for Ruby APIReferencia.

`ListLexicons`

En el siguiente ejemplo de código, se muestra cómo usar `ListLexicons`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 => ex
  puts "Could not get lexicons"
  puts "Error message:"
  puts ex.message
end
```

- Para API obtener más información, consulte ListLexiconsla AWS SDK for Ruby APIReferencia.

`SynthesizeSpeech`

En el siguiente ejemplo de código, se muestra cómo usar `SynthesizeSpeech`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 => ex
  puts "Got error:"
  puts "Error message:"
  puts ex.message
end
```

- Para API obtener más información, consulte SynthesizeSpeechla AWS SDK for Ruby APIReferencia.

Escenarios

Creación de una aplicación para analizar los comentarios de los clientes

El siguiente ejemplo de código muestra cómo crear una aplicación que analice las tarjetas de comentarios de los clientes, las traduzca del idioma original, determine sus opiniones y genere un archivo de audio a partir del texto traducido.

SDKpara Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para obtener el código fuente y las instrucciones de implementación, consulte el proyecto en GitHub.

Servicios utilizados en este ejemplo

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

RDSEjemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonRDS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Hola Amazon RDS

Los siguientes ejemplos de código muestran cómo empezar a utilizar AmazonRDS.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte D escribeDBInstances en la AWS SDK for Ruby APIreferencia.

Temas

- Acciones
- Ejemplos sin servidor

Acciones

`CreateDBSnapshot`

En el siguiente ejemplo de código, se muestra cómo usar `CreateDBSnapshot`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte C reateDBSnapshot en la AWS SDK for Ruby APIreferencia.

`DescribeDBInstances`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeDBInstances`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte D escribeDBInstances en la AWS SDK for Ruby APIreferencia.

`DescribeDBParameterGroups`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeDBParameterGroups`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte escribeDBParameterlos grupos D en AWS SDK for Ruby APIla referencia.

`DescribeDBParameters`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeDBParameters`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte D escribeDBParameters en la AWS SDK for Ruby APIreferencia.

`DescribeDBSnapshots`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeDBSnapshots`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte D escribeDBSnapshots en la AWS SDK for Ruby APIreferencia.

Ejemplos sin servidor

Conexión a una RDS base de datos de Amazon en una función Lambda

El siguiente ejemplo de código muestra cómo implementar una función Lambda que se conecta a una RDS base de datos. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Conexión a una RDS base de datos de Amazon en una función Lambda mediante Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
```

```
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Ejemplos de Amazon S3 que utilizan SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby mediante Amazon S3.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Introducción

Introducción a Amazon S3

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon S3.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte ListBucketsla AWS SDK for Ruby APIReferencia.

Temas

- Conceptos básicos
- Acciones
- Escenarios
- Ejemplos sin servidor

Conceptos básicos

Aprenda los conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un bucket y cargar un archivo en el bucket.
- Descargar un objeto desde un bucket.
- Copiar un objeto en una subcarpeta de un bucket.

- Obtención de una lista de los objetos de un bucket.
- Eliminación del bucket y todos los objetos que incluye.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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: "doc-example-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__
```

- Para API obtener más información, consulte los siguientes temas en AWS SDK for Ruby APIReference.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Acciones

`CopyObject`

En el siguiente ejemplo de código, se muestra cómo usar `CopyObject`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Copie un objeto.

```
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 = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-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__
```

Copie un objeto y añada cifrado del lado del servidor al objeto de destino.

```
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 = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-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__
```

- Para API obtener más información, consulte CopyObjectla AWS SDK for Ruby APIReferencia.

`CreateBucket`

En el siguiente ejemplo de código, se muestra cómo usar `CreateBucket`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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("doc-example-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__
```

- Para API obtener más información, consulte CreateBucketla AWS SDK for Ruby APIReferencia.

`DeleteBucket`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteBucket`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # 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
```

- Para API obtener más información, consulte DeleteBucketla AWS SDK for Ruby APIReferencia.

`DeleteBucketCors`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteBucketCors`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte DeleteBucketCorsla AWS SDK for Ruby APIReferencia.

`DeleteBucketPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteBucketPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteBucketPolicyla AWS SDK for Ruby APIReferencia.

`DeleteObjects`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteObjects`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para API obtener más información, consulte DeleteObjectsla AWS SDK for Ruby APIReferencia.

`GetBucketCors`

En el siguiente ejemplo de código, se muestra cómo usar `GetBucketCors`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 get_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
```

- Para API obtener más información, consulte GetBucketCorsla AWS SDK for Ruby APIReferencia.

`GetBucketPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `GetBucketPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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 get_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
```

- Para API obtener más información, consulte GetBucketPolicyla AWS SDK for Ruby APIReferencia.

`GetObject`

En el siguiente ejemplo de código, se muestra cómo usar `GetObject`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Obtenga un objeto.

```
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 = "doc-example-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__
```

Obtenga un objeto e informe de su estado de cifrado del lado del servidor.

```
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 get_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 = "doc-example-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__
```

- Para API obtener más información, consulte GetObjectla AWS SDK for Ruby APIReferencia.

`HeadObject`

En el siguiente ejemplo de código, se muestra cómo usar `HeadObject`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 = "doc-example-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__
```

- Para API obtener más información, consulte HeadObjectla AWS SDK for Ruby APIReferencia.

`ListBuckets`

En el siguiente ejemplo de código, se muestra cómo usar `ListBuckets`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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__
```

- Para API obtener más información, consulte ListBucketsla AWS SDK for Ruby APIReferencia.

`ListObjectsV2`

En el siguiente ejemplo de código, se muestra cómo usar `ListObjectsV2`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 = "doc-example-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__
```

- Para API obtener más información, consulte ListObjectsV2 en AWS SDK for Ruby APIla referencia.

`PutBucketCors`

En el siguiente ejemplo de código, se muestra cómo usar `PutBucketCors`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte PutBucketCorsla AWS SDK for Ruby APIReferencia.

`PutBucketPolicy`

En el siguiente ejemplo de código, se muestra cómo usar `PutBucketPolicy`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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 set_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
```

- Para API obtener más información, consulte PutBucketPolicyla AWS SDK for Ruby APIReferencia.

`PutBucketWebsite`

En el siguiente ejemplo de código, se muestra cómo usar `PutBucketWebsite`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 = "doc-example-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__
```

- Para API obtener más información, consulte PutBucketWebsitela AWS SDK for Ruby APIReferencia.

`PutObject`

En el siguiente ejemplo de código, se muestra cómo usar `PutObject`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cargue un archivo con un cargador administrado (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 = "doc-example-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__
```

Cargue un archivo con Object.put.

```
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 = "doc-example-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__
```

Cargue un archivo con Object.put y añada cifrado del lado del servidor.

```
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 = "doc-example-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__
```

- Para API obtener más información, consulte PutObjectla AWS SDK for Ruby APIReferencia.

Escenarios

Cree un prefirmado URL

El siguiente ejemplo de código muestra cómo crear un objeto prefirmado URL para Amazon S3 y cómo cargar un objeto.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 = "doc-example-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__
```

Ejemplos sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a Amazon S3 API para recuperar y registrar el tipo de contenido del objeto.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumo de un evento de S3 con Lambda mediante Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

SESEjemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonSES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`GetIdentityVerificationAttributes`

En el siguiente ejemplo de código, se muestra cómo usar `GetIdentityVerificationAttributes`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
  if status == "Success"
    puts email
  end
end
```

- Para API obtener más información, consulte GetIdentityVerificationAttributesla AWS SDK for Ruby APIReferencia.

`ListIdentities`

En el siguiente ejemplo de código, se muestra cómo usar `ListIdentities`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
  if status == "Success"
    puts email
  end
end
```

- Para API obtener más información, consulte ListIdentitiesla AWS SDK for Ruby APIReferencia.

`SendEmail`

En el siguiente ejemplo de código, se muestra cómo usar `SendEmail`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para API obtener más información, consulte SendEmailla AWS SDK for Ruby APIReferencia.

`VerifyEmailIdentity`

En el siguiente ejemplo de código, se muestra cómo usar `VerifyEmailIdentity`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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 => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para API obtener más información, consulte VerifyEmailIdentityla AWS SDK for Ruby APIReferencia.

Ejemplos de Amazon SES API v2 que utilizan SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon SES API v2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`SendEmail`

En el siguiente ejemplo de código, se muestra cómo usar `SendEmail`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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)
```

- Para API obtener más información, consulte SendEmailla AWS SDK for Ruby APIReferencia.

SNSEjemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonSNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones
- Ejemplos sin servidor

Acciones

`CreateTopic`

En el siguiente ejemplo de código, se muestra cómo usar `CreateTopic`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte CreateTopicla AWS SDK for Ruby APIReferencia.

`ListSubscriptions`

En el siguiente ejemplo de código, se muestra cómo usar `ListSubscriptions`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte ListSubscriptionsla AWS SDK for Ruby APIReferencia.

`ListTopics`

En el siguiente ejemplo de código, se muestra cómo usar `ListTopics`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte ListTopicsla AWS SDK for Ruby APIReferencia.

`Publish`

En el siguiente ejemplo de código, se muestra cómo usar `Publish`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte Publicar en AWS SDK for Ruby APIreferencia.

`SetTopicAttributes`

En el siguiente ejemplo de código, se muestra cómo usar `SetTopicAttributes`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte SetTopicAttributesla AWS SDK for Ruby APIReferencia.

`Subscribe`

En el siguiente ejemplo de código, se muestra cómo usar `Subscribe`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Suscribe una dirección de correo electrónico a un tema.

```
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
```

- Para obtener información, consulte la Guía para desarrolladores de AWS SDK for Ruby.
- Para API obtener más información, consulte Suscríbete AWS SDK for Ruby APIcomo referencia.

Ejemplos sin servidor

Invocar una función Lambda desde un disparador de Amazon SNS

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de mensajes de un SNS tema. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un SNS evento con Lambda mediante Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

SQSEjemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con AmazonSQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`ChangeMessageVisibility`

En el siguiente ejemplo de código, se muestra cómo usar `ChangeMessageVisibility`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })
```

```
  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 '#{receive_queue_name}', as it
 does not exist."
end
```

- Para API obtener más información, consulte ChangeMessageVisibilityla AWS SDK for Ruby APIReferencia.

`CreateQueue`

En el siguiente ejemplo de código, se muestra cómo usar `CreateQueue`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# 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__
```

- Para API obtener más información, consulte CreateQueuela AWS SDK for Ruby APIReferencia.

`DeleteQueue`

En el siguiente ejemplo de código, se muestra cómo usar `DeleteQueue`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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)
```

- Para API obtener más información, consulte DeleteQueuela AWS SDK for Ruby APIReferencia.

`ListQueues`

En el siguiente ejemplo de código, se muestra cómo usar `ListQueues`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte ListQueuesla AWS SDK for Ruby APIReferencia.

`ReceiveMessage`

En el siguiente ejemplo de código, se muestra cómo usar `ReceiveMessage`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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__
```

- Para API obtener más información, consulte ReceiveMessagela AWS SDK for Ruby APIReferencia.

`SendMessage`

En el siguiente ejemplo de código, se muestra cómo usar `SendMessage`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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__
```

- Para API obtener más información, consulte SendMessagela AWS SDK for Ruby APIReferencia.

`SendMessageBatch`

En el siguiente ejemplo de código, se muestra cómo usar `SendMessageBatch`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
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
```

- Para API obtener más información, consulte SendMessageBatchla AWS SDK for Ruby APIReferencia.

Ejemplos sin servidor

Invocar una función Lambda desde un disparador de Amazon SQS

El siguiente ejemplo de código muestra cómo implementar una función Lambda que recibe un evento desencadenado por la recepción de mensajes de una SQS cola. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Consumir un SQS evento con Lambda mediante Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Notificación de errores de artículos de lotes para funciones de Lambda con un activador de Amazon SQS

El siguiente ejemplo de código muestra cómo implementar una respuesta por lotes parcial para las funciones de Lambda que reciben eventos de una SQS cola. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de ejemplos sin servidor.

Informe de errores de elementos de SQS lote con Lambda mediante Ruby.

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

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS ejemplos que utilizan SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK for Ruby with AWS STS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`AssumeRole`

En el siguiente ejemplo de código, se muestra cómo usar `AssumeRole`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # 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
```

- Para API obtener más información, consulte AssumeRolela AWS SDK for Ruby APIReferencia.

Ejemplos de Amazon Textract usando Ruby SDK

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby con Amazon Textract.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Escenarios

Escenarios

Creación de una aplicación para analizar los comentarios de los clientes

El siguiente ejemplo de código muestra cómo crear una aplicación que analice las tarjetas de comentarios de los clientes, las traduzca del idioma original, determine sus opiniones y genere un archivo de audio a partir del texto traducido.

SDKpara Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para obtener el código fuente y las instrucciones de implementación, consulte el proyecto en GitHub.

Servicios utilizados en este ejemplo

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

Ejemplos de Amazon Translate que utilizan SDK Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK for Ruby mediante Amazon Translate.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Escenarios

Escenarios

Creación de una aplicación para analizar los comentarios de los clientes

El siguiente ejemplo de código muestra cómo crear una aplicación que analice las tarjetas de comentarios de los clientes, las traduzca del idioma original, determine sus opiniones y genere un archivo de audio a partir del texto traducido.

SDKpara Ruby

Esta aplicación de ejemplo analiza y almacena las tarjetas de comentarios de los clientes. Concretamente, satisface la necesidad de un hotel ficticio en la ciudad de Nueva York. El hotel recibe comentarios de los huéspedes en varios idiomas en forma de tarjetas de comentarios físicas. Esos comentarios se cargan en la aplicación a través de un cliente web. Una vez cargada la imagen de una tarjeta de comentarios, se llevan a cabo los siguientes pasos:

- El texto se extrae de la imagen mediante Amazon Textract.
- Amazon Comprehend determina la opinión del texto extraído y su idioma.
- El texto extraído se traduce al inglés mediante Amazon Translate.
- Amazon Polly sintetiza un archivo de audio a partir del texto extraído.

La aplicación completa se puede implementar con AWS CDK. Para obtener el código fuente y las instrucciones de implementación, consulte el proyecto en GitHub.

Servicios utilizados en este ejemplo

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

WorkDocs Ejemplos de Amazon que utilizan SDK Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK for Ruby con Amazon WorkDocs.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`DescribeRootFolders`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeRootFolders`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
  # Retrieves the root folder for a user by email
  # @param users [Array<Types::User>] A list of users selected from API response
  # @param user_email [String] The email of the user.
  def get_user_folder(users, user_email)
    user = users.find { |user| user.email_address == user_email }
    if user
      user.root_folder_id
    else
      @logger.error "Could not get root folder for user with email address
#{user_email}"
```

```
      exit(1)
    end
  end

  # Describes the contents of a folder
  # @param [String] folder_id - The Id of the folder to describe.
  def describe_folder_contents(folder_id)
    resp = @client.describe_folder_contents({
                                              folder_id: folder_id, # required
                                              sort: "NAME", # accepts DATE, NAME
                                              order: "ASCENDING", # accepts
ASCENDING, DESCENDING
                                            })
    resp.documents.each do |doc|
      md = doc.latest_version_metadata
      @logger.info "Name:          #{md.name}"
      @logger.info "Size (bytes):  #{md.size}"
      @logger.info "Last modified: #{doc.modified_timestamp}"
      @logger.info "Doc ID:        #{doc.id}"
      @logger.info "Version ID:    #{md.id}"
      @logger.info ""
    end
  rescue Aws::WorkDocs::Errors::ServiceError => e
    @logger.error "Error listing folder contents: #{e.message}"
    exit(1)
  end
```

- Para API obtener más información, consulte DescribeRootFolderslа AWS SDK for Ruby APIReferencia.

`DescribeUsers`

En el siguiente ejemplo de código, se muestra cómo usar `DescribeUsers`.

SDKpara Ruby

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
 # Describes users within an organization
 # @param [String] org_id: The ID of the org.
 def describe_users(org_id)
   resp = @client.describe_users({
                                   organization_id: org_id,
                                   include: "ALL", # accepts ALL, ACTIVE_PENDING
                                   order: "ASCENDING", # accepts ASCENDING,
DESCENDING
                                   sort: "USER_NAME", # accepts USER_NAME,
FULL_NAME, STORAGE_LIMIT, USER_STATUS, STORAGE_USED
                                 })
   resp.users.each do |user|
     @logger.info "First name:  #{user.given_name}"
     @logger.info "Last name:   #{user.surname}"
     @logger.info "Email:       #{user.email_address}"
     @logger.info "Root folder: #{user.root_folder_id}"
     @logger.info ""
   end
   resp.users
 rescue Aws::WorkDocs::Errors::ServiceError => e
   @logger.error "AWS WorkDocs Service Error: #{e.message}"
   exit(1)
 end
```

- Para API obtener más información, consulte DescribeUsersla AWS SDK for Ruby APIReferencia.

Seguridad AWS SDK para Ruby

La seguridad en la nube de Amazon Web Services (AWS) es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre usted AWS y usted. En el modelo de responsabilidad compartida, se habla de “seguridad de la nube” y “seguridad en la nube”:

Seguridad de la nube: AWS se encarga de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la AWS nube y de proporcionarle servicios que pueda utilizar de forma segura. Nuestra responsabilidad en materia de seguridad es nuestra máxima prioridad AWS, y auditores externos comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS conformidad.

Seguridad en la nube: su responsabilidad viene determinada por el uso Servicio de AWS que utilice y por otros factores, como la confidencialidad de sus datos, los requisitos de su organización y las leyes y reglamentos aplicables.

Temas

La protección de datos es AWS SDK para Ruby

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información

sobre la privacidad de los datos, consulte la sección Privacidad de datos FAQ. Para obtener información sobre la protección de datos en Europa, consulte el modelo de responsabilidad AWS compartida y la entrada del GDPR blog sobre AWS seguridad.

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactorial (MFA) con cada cuenta.
- UseSSL/TLSpara comunicarse con AWS los recursos. Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Configure API y registre la actividad del usuario con AWS CloudTrail.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita entre FIPS 140 y 3 módulos criptográficos validados para acceder a AWS través de una interfaz de línea de comandos o unaAPI, utilice un FIPS terminal. Para obtener más información sobre los FIPS puntos finales disponibles, consulte la Norma federal de procesamiento de información () FIPS 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con o Servicios de AWS utiliza la consola, API AWS CLI, o. AWS SDKs Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, le recomendamos encarecidamente que no incluya la información sobre las credenciales URL para validar la solicitud a ese servidor.

Identity and Access Management AWS SDK para Ruby

AWS Identity and Access Management (IAM) es un servicio de Amazon Web Services (AWS) que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. IAMlos administradores controlan quién puede autenticarse (iniciar sesión) y quién está autorizado (tiene

permisos) para usar los recursos Servicios de AWS. IAMes una Servicio de AWS que puede utilizar sin coste adicional.

Para poder acceder AWS SDK a Ruby AWS, necesitas una AWS cuenta y unas AWS credenciales. Para aumentar la seguridad de tu AWS cuenta, te recomendamos que utilices un IAMusuario para proporcionar las credenciales de acceso en lugar de utilizar las credenciales de tu AWS cuenta.

Para obtener más información sobre cómo trabajar conIAM, consulte IAM.

Para obtener información general sobre los IAM usuarios y por qué son importantes para la seguridad de su cuenta, consulte Credenciales de AWS seguridad en la Referencia general de Amazon Web Services.

AWS SDKfor Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Validación de conformidad AWS SDK para Ruby

AWS SDKfor Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Auditores externos evalúan la seguridad y la conformidad de los servicios de Amazon Web Services (AWS) como parte de varios programas de AWS conformidad. Estos incluyen SOC PCI la Reserva Federal RAMP HIPAA y otros. AWS proporciona una lista actualizada con frecuencia del alcance de Servicios de AWS los programas de cumplimiento específicos en AWS Services in Scope by Compliance Program.

Los informes de auditoría de terceros están disponibles para que los descargue y los utilice AWS Artifact. Para obtener más información, consulta Descarga de informes en AWS Artifact.

Para obtener más información sobre los programas AWS de conformidad, consulte Programas AWS de conformidad.

Tu responsabilidad en materia de cumplimiento cuando utilices Ruby AWS SDK para acceder a un sitio Servicio de AWS está determinada por la confidencialidad de tus datos, los objetivos de cumplimiento de tu organización y las leyes y reglamentos aplicables. Si el uso que haces de un Servicio de AWS sitio está sujeto al cumplimiento de normas como HIPAA las de la Reserva FederalRAMP, te ofrecemos AWS recursos que te ayudarán a: PCI

- Guías de inicio rápido sobre seguridad y conformidad: guías de implementación en las que se analizan las consideraciones arquitectónicas y se proporcionan los pasos necesarios para implementar entornos básicos centrados en la seguridad y el cumplimiento. AWS
- Documento técnico sobre la arquitectura basada en HIPAA la seguridad y el cumplimiento: un documento técnico que describe cómo las empresas pueden utilizar para crear aplicaciones que cumplan con los requisitos. AWS HIPAA
- AWS Recursos de cumplimiento: una colección de libros de trabajo y guías que pueden aplicarse a su sector y ubicación.
- AWS Config: un servicio que evalúa en qué medida las configuraciones de sus recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub: una visión integral del estado de su seguridad AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.

Resiliencia AWS SDK para Ruby

La infraestructura global de Amazon Web Services (AWS) se basa en Regiones de AWS zonas de disponibilidad.

Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia.

Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

AWS SDKfor Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS

de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Seguridad de infraestructura AWS SDK para Ruby

AWS SDKfor Ruby sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información Servicio de AWS de seguridad, consulte la página Servicio de AWS de documentación de seguridad y Servicios de AWS consulte el alcance de las iniciativas de AWS cumplimiento implementadas por el programa de cumplimiento.

Para obtener información sobre los procesos AWS de seguridad, consulte el documento técnico AWS: Descripción general de los procesos de seguridad.

Imponer una TLS versión mínima en el para Ruby AWS SDK

La AWS SDK comunicación entre Ruby y AWS está asegurada mediante Secure Sockets Layer (SSL) o Transport Layer Security (TLS). Todas las SSL versiones y versiones TLS anteriores a la 1.2 tienen vulnerabilidades que pueden comprometer la seguridad de la comunicación AWS. Por este motivo, debes asegurarte de usar la versión AWS SDK para Ruby con una versión de Ruby compatible con la TLS versión 1.2 o posterior.

Ruby usa la SSL biblioteca Open para proteger HTTP las conexiones. Las versiones compatibles de Ruby (1.9.3 y posteriores) que se instalan mediante administradores de paquetes del sistema (`yum`apt, y otros), un instalador oficial o administradores de Ruby (rbenv y otros) suelen incorporar Open SSL 1.0.1 o posterior, que es compatible con 1.2. RVM TLS

Cuando se usa con una versión compatible de Ruby con Open SSL 1.0.1 o posterior, la versión AWS SDK para Ruby prefiere la TLS 1.2 y utiliza la versión más reciente SSL o TLS compatible tanto con el cliente como con el servidor. Siempre es al menos TLS 1.2 para Servicios de AWS. (SDKUsa la `Net::HTTP` clase Ruby con`use_ssl=true`.)

Comprobando la SSL versión abierta

Para asegurarte de que tu instalación de Ruby usa Open SSL 1.0.1 o una versión posterior, ingresa el siguiente comando.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Una forma alternativa de obtener la SSL versión abierta es consultar directamente el `openssl` ejecutable. Primero, localice el ejecutable adecuado mediante el siguiente comando.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

El resultado debería `--with-openssl-dir=/path/to/openssl` indicar la ubicación de la SSL instalación abierta. Apunte esta ruta. Para comprobar la versión de OpenSSL, introduzca los siguientes comandos.

```
cd /path/to/openssl
bin/openssl version
```

Este último método puede no funcionar en todas las instalaciones de Ruby.

TLSSoporte de actualización

Si la versión de Open que SSL utiliza su instalación de Ruby es anterior a la 1.0.1, actualice su SSL instalación de Ruby o Open mediante el administrador de paquetes del sistema, el instalador de Ruby o el administrador de Ruby, tal y como se describe en la guía de instalación de Ruby. Si instalas Ruby desde el código fuente, instala SSL primero la versión más reciente de Open y luego pásala `--with-openssl-dir=/path/to/upgraded/openssl` cuando la estés ejecutando `./configure`.

Migración de Amazon S3 Encryption Client

En este tema se muestra cómo migrar las aplicaciones de la versión 1 (V1) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) a la versión 2 (V2) y cómo garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración.

Información general sobre la migración

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. En primer lugar, implemente una versión actualizada de AWS SDK para Ruby en su aplicación. Así, permitirá a los clientes de cifrado de la versión V1 descifrar los objetos escritos por los nuevos clientes de la versión V2. Si su aplicación usa varias AWS SDKs, debe actualizar cada una SDK por separado.

2. Migue los clientes de cifrado y descifrado a la versión V2. Una vez que todos sus clientes de cifrado de la versión 1 puedan leer los nuevos formatos, puede migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones de la versión 2.

Actualizar los clientes existentes para leer nuevos formatos

El cliente de cifrado de la versión V2 utiliza algoritmos de cifrado que las versiones anteriores del cliente no admiten. El primer paso de la migración consiste en actualizar los clientes de descifrado de la versión 1 a la SDK versión más reciente. Tras completar este paso, los clientes de la versión V1 de su aplicación podrán descifrar los objetos cifrados por los clientes de cifrado de la versión V2. Consulte los detalles a continuación para ver cada versión principal AWS SDK de Ruby.

Actualización AWS SDK para la versión 3 de Ruby

La versión 3 es la última versión de AWS SDK For Ruby. Para completar la migración, debe utilizar la versión 1.76.0 o una posterior de la gema `aws-sdk-s3`.

Instalación desde la línea de comandos

Para los proyectos que instalan la gema `aws-sdk-s3`, utilice la opción de versión para comprobar que está instalada la versión mínima de 1.76.0.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Uso de archivos Gemfile

Establezca la versión mínima de la gema `aws-sdk-s3` en 1.76.0 para los proyectos que utilizan un archivo Gemfile para gestionar las dependencias. Por ejemplo:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Modifique su archivo Gemfile.
2. Ejecute `bundle update aws-sdk-s3`. Para comprobar su versión, ejecute `bundle info aws-sdk-s3`.

Actualización para la versión 2 AWS SDK de Ruby

La versión 2 AWS SDK para Ruby entrará en modo de mantenimiento el 21 de noviembre de 2021. Para completar la migración, debe utilizar la versión 2.11.562 o una posterior de la gema aws-sdk.

Instalación desde la línea de comandos

Para los proyectos que instalan la gema `aws-sdk`, utilice la opción de versión para comprobar que está instalada la versión mínima de 2.11.562.

```
gem install aws-sdk -v '>= 2.11.562'
```

Uso de archivos Gemfile

Establezca la versión mínima de la gema `aws-sdk` en 2.11.562 para los proyectos que utilizan un archivo Gemfile para gestionar las dependencias. Por ejemplo:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Modifique su archivo Gemfile. Si tiene un archivo Gemfile.lock, elimínelo o actualícelo.
2. Ejecute `bundle update aws-sdk`. Para comprobar su versión, ejecute `bundle info aws-sdk`.

Migrar clientes de cifrado y descifrado a la versión V2

Después de actualizar sus clientes para leer los nuevos formatos de cifrado, puede actualizar sus aplicaciones a la versión V2 de los clientes de cifrado y descifrado. En los siguientes pasos, se muestra cómo migrar correctamente el código de la versión V1 a la V2.

Antes de actualizar el código para usar el cliente de cifrado V2, siga los pasos anteriores y utilice la gema `aws-sdk-s3` en la versión 2.11.562 o posterior.

> **Note**
>
> Al descifrar con AES -GCM, lee todo el objeto hasta el final antes de empezar a usar los datos descifrados. Esto se hacer para verificar que el objeto no se ha modificado desde que se cifró.

Configuración de clientes de cifrado de la versión V2

El EncryptionV2::Client requiere una configuración adicional. Para obtener información de configuración detallada, consulte la documentación de EncryptionV2::Client o los ejemplos que se proporcionan más adelante en este tema.

1. El método de encapsulación de clave y el algoritmo de cifrado del contenido deben especificarse al construir el cliente. Al crear un `EncryptionV2::Client` nuevo, debe proporcionar valores para `key_wrap_schema` y `content_encryption_schema`.

`key_wrap_schema`- Si está utilizando AWS KMS, debe estar configurado en. `:kms_context` Si utiliza una clave simétrica (AES), debe estar configurada en. `:aes_gcm` Si utiliza una clave asimétrica (RSA), debe estar configurada en. `:rsa_oaep_sha1`

`content_encryption_schema` - Este se debe establecer en aes_gcm_no_padding.

2. security_profile debe especificarse en la construcción del cliente. Al crear un `EncryptionV2::Client` nuevo, debe proporcionar un valor para `security_profile`. El parámetro security_profile determina la compatibilidad para los objetos de lectura escritos con la versión V1 de `Encryption::Client` anterior. Hay dos valores: :v2 y :v2_and_legacy. Para admitir la migración, establezca el `security_profile` en :v2_and_legacy. Use :v2 solo para el desarrollo de nuevas aplicaciones.

3. AWS KMS key se aplica de forma predeterminada. En la versión 1`Encryption::Client`, la `kms_key_id` utilizada para crear el cliente no se proporcionó a. AWS KMS `Decrypt call` AWS KMS puede obtener esta información de los metadatos y añadirla al blob simétrico de texto cifrado. En la versión 2, e`EncryptionV2: :Client`, el kms_key_id se pasa a la llamada Decrypt y la llamada falla si no coincide con la clave utilizada para AWS KMS cifrar el objeto. Si anteriormente no estableció un `kms_key_id` específico para su código, establezca `kms_key_id: :kms_allow_decrypt_with_any_cmk` en la creación del cliente o establezca `kms_allow_decrypt_with_any_cmk: true` en llamadas de `get_object`.

Ejemplo: usar AES una clave simétrica ()

Antes de la migración

```
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)
```

Después de la migración

```
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
```

Ejemplo: usar AWS KMS con kms_key_id

Antes de la migración

```
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)
```

Después de la migración

```
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
```

Ejemplo: usar sin kms_key_id AWS KMS

Antes de la migración

```
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)
```

Después de la migración

```
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
```

Alternativa posterior a la migración

Si solo lee y descifra (nunca escribe ni cifra) objetos con el cliente de cifrado de S2, utilice este código.

```
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
```

Historial de documentos

En la siguiente tabla se describen cambios importantes en esta guía. Para recibir notificaciones sobre las actualizaciones de esta documentación, puedes suscribirte a un RSSfeed.

Cambio	Descripción	Fecha
Tabla de contenido y ejemplos guiados	Se eliminaron los ejemplos guiados para pasarlos al repositorio de ejemplos de código, que es más completo.	10 de julio de 2024
Índice	Se ha actualizado el índice para que los ejemplos de código sean más accesibles.	1 de junio de 2023
IAMactualizaciones de mejores prácticas	Guía actualizada para adaptarla a las IAM mejores prácticas. Para obtener más información, consulte las mejores prácticas de seguridad en IAM. Se ha actualizado el tutorial de introducción.	8 de mayo de 2023
Actualizaciones generales	Se ha actualizado la página de bienvenida de los recursos externos pertinentes. También se ha actualizado la versión mínima requerida de Ruby para la versión 2.3. AWS Key Management Service Secciones actualizadas para reflejar las actualizaciones terminológicas. Se actualizó la información de uso de la	8 de agosto de 2022

	REPL utilidad para mayor claridad.	
Corrección de enlaces que no funcionaban	Se han corregido enlaces que no funcionaban. Se eliminó la página redundante de consejos y trucos; se redirigió al contenido de EC2 ejemplo de Amazon. Se incluyen listas de los ejemplos de código que están disponibles GitHub en el repositorio de ejemplos de código.	3 de agosto de 2022
SDKMétricas	Se ha eliminado la informaci ón sobre la activación de SDK Metrics for Enterprise Support, que ha quedado obsoleta.	28 de enero de 2022

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.