

Guía para desarrolladores

AWS SDK for Ruby



AWS SDK for Ruby: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es AWS SDK para Ruby?	1
Documentación y recursos adicionales	1
Implementación en la nube de AWS	1
Mantenimiento y compatibilidad de las versiones principales del SDK	2
Introducción	3
Autenticarse con AWS	3
Uso de credenciales de consola	3
Uso de la autenticación del Centro de Identidad de IAM	3
Información adicional de autenticación	5
Instalación del SDK	6
Requisitos previos	6
Instalación del SDK	6
Creación de una aplicación simple	8
Escritura del código	8
Ejecución del programa	9
Nota para usuarios de Windows	9
Pasos a seguir a continuación	10
Configuración de clientes de servicio	11
Prioridad de los ajustes	12
Configuración externa de clientes	12
Variables de entorno del AWS SDK para Ruby	14
Configuración de clientes en código	14
Aws.config	14
Región de AWS	16
Orden de búsqueda de regiones para la resolución	16
Cómo configurar la región	16
Proveedores de credencial	18
Cadena de proveedores de credenciales	18
Crear un token de AWS STS acceso	20
Reintentos	21
Especificación del comportamiento de reintento del cliente en el código	21
Observabilidad	22
Configuración de un OTelProvider para un cliente de servicio	22
Configuración de un OTelProvider para todos los clientes de servicio	25

Configuración de un proveedor de telemetría personalizado	25
Atributos del intervalo	26
HTTP	28
Configuración de un punto de conexión no estándar	28
Uso del SDK	29
Realización de solicitudes de Servicio de AWS	29
Uso de la utilidad REPL	30
Requisitos previos	30
Configuración de Bundler	31
Ejecución de REPL	31
Uso del SDK con Ruby on Rails	32
Depuración mediante rastreo de red de un cliente	32
Probando con stubbing	33
Disociación de respuestas de cliente	34
Disociación de errores de cliente	35
Paginación	35
Las respuestas paginadas se pueden enumerar	35
Administrar respuestas paginadas manualmente	36
Clases de datos paginados	36
Esperadores	37
Invocación de un esperador	37
Errores de espera	37
Configuración de un esperador	38
Ampliación de un esperador	38
Ejemplos de código	40
Aurora	41
Introducción	41
escalado automático	42
Introducción	41
CloudTrail	44
Acciones	44
CloudWatch	49
Acciones	44
Amazon Cognito Identity Provider	61
Introducción	41
Amazon Comprehend	63

Escenarios	63
Amazon DocumentDB	64
Ejemplos de tecnología sin servidor	65
DynamoDB	66
Introducción	41
Conceptos básicos	68
Acciones	44
Escenarios	63
Ejemplos de tecnología sin servidor	65
Amazon EC2	95
Introducción	41
Acciones	44
Elastic Beanstalk	129
Acciones	44
EventBridge	135
Escenarios	63
AWS Glue	153
Introducción	41
Conceptos básicos	68
Acciones	44
IAM	181
Introducción	41
Conceptos básicos	68
Acciones	44
Kinesis	240
Ejemplos de tecnología sin servidor	65
AWS KMS	242
Acciones	44
Lambda	246
Introducción	41
Conceptos básicos	68
Acciones	44
Escenarios	63
Ejemplos de tecnología sin servidor	65
Amazon MSK	276
Ejemplos de tecnología sin servidor	65

Amazon Polly	277
Acciones	44
Escenarios	63
Amazon RDS	282
Introducción	41
Acciones	44
Ejemplos de tecnología sin servidor	65
Amazon S3	290
Introducción	41
Conceptos básicos	68
Acciones	44
Escenarios	63
Ejemplos de tecnología sin servidor	65
Amazon SES	322
Acciones	44
Amazon SES API v2	328
Acciones	44
Amazon SNS	329
Acciones	44
Ejemplos de tecnología sin servidor	65
Amazon SQS	339
Acciones	44
Ejemplos de tecnología sin servidor	65
AWS STS	353
Acciones	44
Amazon Textract	354
Escenarios	63
Amazon Translate	355
Escenarios	63
Migración de versiones	357
Uso en paralelo	357
Diferencias generales	357
Diferencias del cliente	358
Diferencias en los recursos	359
Seguridad	361
Protección de los datos	361

Gestión de identidad y acceso	363
Validación de la conformidad	363
Resiliencia	364
Seguridad de infraestructuras	365
Aplicación de una versión mínima de TLS	365
Comprobar la versión de OpenSSL	366
Actualizar la compatibilidad con TLS	366
Migración del cliente de cifrado S3 (V1 a V2)	366
Información general sobre la migración	367
Actualizar los clientes existentes para leer nuevos formatos	367
Migrar clientes de cifrado y descifrado a la versión V2	368
Migración del cliente de cifrado S3 (V2 a V3)	372
Información general sobre la migración	372
Comprensión de las funciones de la V3	373
Actualizar los clientes existentes para leer nuevos formatos	376
Migre los clientes de cifrado y descifrado a la V3	377
Historial de documentos	385
.....	ccclxxxvii

¿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 al 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 para 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](#)

Implementación en la nube de AWS

Puede utilizar los Servicios de AWS, por ejemplo, AWS Elastic Beanstalk y AWS CodeDeploy para implementar la 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 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](#):

- [AWS Política de mantenimiento de SDK y herramientas](#)
- [AWS Matriz de compatibilidad para versiones de SDK y herramientas](#)

Introducción al 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 con AWS SDK for Ruby](#)
- [Instalación de AWS SDK para Ruby](#)
- [Creación de una aplicación simple mediante el AWS SDK para Ruby](#)

Autenticación con AWS SDK for Ruby

Debes establecer cómo se autentica tu código AWS al desarrollar con Servicios de AWS. Puedes configurar el acceso programático a AWS los recursos de diferentes maneras en función del entorno y del AWS acceso del que dispongas.

Para elegir el método de autenticación y configurarlo para el SDK, consulte [Autenticación y acceso](#) en la Guía de referencia sobre herramientas AWS SDKs y herramientas.

Uso de credenciales de consola

Para el desarrollo local, recomendamos que los nuevos usuarios utilicen sus credenciales de inicio de sesión AWS de Management Console existentes para acceder a AWS los servicios mediante programación. Tras la autenticación basada en el navegador, AWS genera credenciales temporales que funcionan con herramientas de desarrollo locales, como la interfaz de línea de AWS comandos (AWS CLI) y el AWS SDK for Ruby.

Si elige este método, siga las instrucciones [para iniciar sesión para el desarrollo AWS local con las credenciales de la consola mediante la AWS CLI](#).

El AWS SDK para Ruby no necesita añadir gemas adicionales (por ejemplo `aws-sdk-signin`) a tu aplicación para usar el inicio de sesión con credenciales de consola.

Uso de la autenticación del Centro de Identidad de IAM

Si elige este método, complete el procedimiento de [autenticación del Centro de Identidad de IAM](#) que se indica en la Guía de referencia de herramientas AWS SDKs y herramientas. Una vez completado, el entorno debe contener los siguientes elementos:

- El AWS CLI, que se utiliza para iniciar una sesión en el portal de AWS acceso 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 AWS SDKs and Tools.
- El archivo compartido de config establece la configuración de [region](#). Esto establece el valor predeterminado Región de AWS que el SDK usa para AWS las solicitudes. 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 sso_role_name valor, que es un rol de IAM conectado a un conjunto de permisos del Centro de Identidad 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 sso-session sección contiene la configuración para iniciar una sesión en el portal de AWS acceso.

```
[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
```

El AWS SDK para Ruby no necesita añadir gemas adicionales (como aws-sdk-sso y aws-sdk-ssooidc) a la aplicación para utilizar la autenticación del IAM Identity Center.

Inicie una sesión en el portal de AWS acceso

Antes de ejecutar una aplicación para acceder Servicios de AWS, necesita una sesión activa en el portal de AWS acceso para que el SDK utilice la autenticación del IAM Identity Center a fin

de resolver las credenciales. En función de la duración de las sesiones configuradas, el acceso terminará por caducar y SDK detectará un error de autenticación. Para iniciar sesión en el portal de AWS acceso, ejecute el siguiente comando en 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 Centro de identidades de IAM configurados en el archivo `config` compartido.

Note

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

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 sus AWS entornos y aplicaciones. 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 AWS. Puedes usar un proveedor de identidad para que tus usuarios humanos proporcionen acceso federado a AWS las cuentas asumiendo funciones, que proporcionan credenciales temporales. Si desea administrar el acceso de manera centralizada, se recomienda utilizar (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 AWS credenciales de corta duración, consulte [Credenciales de seguridad temporales](#) en la Guía del usuario de IAM.
- Para obtener más información sobre la cadena de proveedores de credenciales de AWS SDK for Ruby y cómo el SDK intenta utilizar automáticamente diferentes métodos de autenticación en una secuencia, consulte [Cadena de proveedores de credenciales](#).
- Para ver los ajustes de configuración del proveedor de credenciales del AWS SDK, consulta los [proveedores de credenciales estandarizados](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

Instalación de 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 con AWS](#) [AWS SDK for Ruby](#).

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 2.5.

```
$ 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
```

Creación de una aplicación simple mediante el 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.

Escritura del código

Copie y pegue el siguiente código en un nuevo archivo de origen. Asigne el nombre al 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.

Pasos a seguir a continuación

Para probar muchas otras operaciones de Amazon S3, consulte el [Repositorio de ejemplos de código de AWS](#) en GitHub.

Configuración de clientes de servicio en el AWS SDK para Ruby

Para acceder mediante programación a los Servicios de AWS, el AWS SDK para Ruby utiliza una clase de cliente para cada Servicio de AWS. Por ejemplo, si su aplicación necesita acceder a Amazon EC2, su aplicación crearía un objeto de cliente de Amazon EC2 para interactuar con ese servicio. A continuación, utiliza el cliente de servicio para realizar solicitudes al mismo Servicio de AWS.

Para realizar una solicitud a un Servicio de AWS, debe crear primero un cliente de servicio. Para cada Servicio de AWS que utilice el código, tiene su propia gema y su propio tipo específico para interactuar con él. El cliente expone un método para cada operación de la API expuesta por el servicio.

Hay muchas maneras alternativas de configurar el comportamiento del SDK, pero, en última instancia, todo está relacionado con el comportamiento de los clientes de servicio. Una configuración no tiene efecto hasta que se utiliza un cliente de servicio creado a partir de ella.

Debe establecer cómo se autentica el código con AWS cuando desarrolla con Servicios de AWS. También debe configurar la Región de AWS en la que lo desea usar.

La [Guía de referencia de las herramientas y los SDK de AWS](#) también contiene configuraciones, características y otros conceptos fundamentales comunes a muchos de los SDK de AWS.

Temas

- [Prioridad de los ajustes](#)
- [Configuración de clientes de servicio de AWS SDK para Ruby de forma externa](#)
- [Configuración de clientes de servicio de AWS SDK para Ruby en código](#)
- [Configuración de la Región de AWS para el AWS SDK para Ruby](#)
- [Uso de AWS SDK para proveedores de credenciales de Ruby](#)
- [Configuración de reintentos en el AWS SDK para Ruby](#)
- [Configuración de las características de observabilidad en AWS SDK para Ruby](#)
- [Configuración de los ajustes por HTTP en el AWS SDK para Ruby](#)

Los [archivos de config y credentials compartidos](#) se pueden usar para los ajustes de configuración. Para ver todos los ajustes del AWS SDK, consulte la [referencia sobre los ajustes](#) en la Guía de referencia de herramientas y AWS SDK.

Se pueden usar diferentes perfiles para almacenar diferentes configuraciones. Para especificar el perfil activo que carga el SDK, puede usar la variable de entorno `AWS_PROFILE` o la opción de `profile` de `Aws.config`.

Prioridad de los ajustes

Los ajustes globales configuran las funciones, los proveedores de credenciales y otras funcionalidades compatibles con la mayoría de los SDK y que tienen un amplio impacto en todos los Servicios de AWS. Todos los SDK de AWS tienen una serie de lugares (u orígenes) que se comprueban para encontrar un valor para la configuración global. No todas las configuraciones están disponibles en todos los orígenes. La siguiente es la configuración de la prioridad de búsqueda:

1. Cualquier ajuste explícito establecido en el código o en el propio cliente de un servicio tiene prioridad sobre cualquier otra cosa.
 - a. Todos los parámetros que se pasen directamente al constructor de un cliente tienen la máxima prioridad.
 - b. `Aws.config` se comprueba para la configuración global o específica del servicio.
2. Se comprueba la variable de entorno .
3. El archivo de `credentials` de AWS compartido está comprobado.
4. El archivo de `config` de AWS compartido está comprobado.
5. Los valores predeterminados proporcionados por el código de origen del AWS SDK para Ruby se utilizan en último lugar.

Configuración de clientes de servicio de AWS SDK para Ruby de forma externa

Muchas opciones de configuración se pueden gestionar fuera del código. Cuando la configuración se gestiona de manera externa, se aplica a todas las aplicaciones. La mayoría de las opciones de configuración se pueden establecer como variables de entorno o en un archivo `config` de AWS independiente compartido. El archivo `config` compartido puede mantener conjuntos de opciones

independientes, llamados perfiles, para proporcionar diferentes configuraciones para distintos entornos o pruebas.

Las variables de entorno y la configuración del archivo `config` compartido están estandarizadas y se comparten entre los AWS SDK y las herramientas para garantizar una funcionalidad coherente en los diferentes lenguajes de programación y aplicaciones.

Consulte la Guía de referencia de herramientas y AWS SDK para obtener información sobre cómo configurar la aplicación con estos métodos, además de obtener información sobre cada ajuste entre SDK. Para ver toda la configuración que el SDK puede resolver a partir de las variables de entorno o los archivos de configuración, consulte [Referencia de configuración](#) en la Guía de referencia de herramientas y AWS SDK.

Para realizar una solicitud a un Servicio de AWS, primero debe crear una instancia de un cliente para ese servicio. Puede configurar los ajustes comunes para los clientes del servicio, como los tiempos de espera, el cliente HTTP y la configuración de reintentos.

Cada cliente de servicio requiere una Región de AWS y un proveedor de credenciales. El SDK usa estos valores para enviar solicitudes a la región correcta de los recursos y para firmar las solicitudes con las credenciales correctas. Puede especificar estos valores mediante programación en el código o hacer que se carguen automáticamente desde el entorno.

El SDK tiene una serie de lugares (u orígenes) que comprueba para encontrar un valor para la configuración global.

1. Cualquier ajuste explícito establecido en el código o en el propio cliente de un servicio tiene prioridad sobre cualquier otra cosa.
2. Variables de entorno
 - Para obtener información sobre cómo establecer variables de entorno, consulte [variables de entorno](#) en la Guía de referencia de herramientas y SDK de AWS.
 - Tenga en cuenta que puede configurar variables de entorno para un intérprete de comandos en diferentes niveles de ámbito: todo el sistema, todos los usuarios y una sesión de terminal específica.
3. Archivos `config` y `credentials` compartidos
 - Para obtener información sobre cómo configurar estos archivos, consulte [Archivos config y credentials compartidos](#) en la Guía de referencia de herramientas y AWS SDK.
4. Los valores predeterminados proporcionados por el código de origen del SDK se utilizan en último lugar.

- Algunas propiedades, como Región, no tienen un valor predeterminado. Debe especificarlas de manera explícita en el código, en la configuración del entorno o en el archivo `config` compartido. Si el SDK no puede resolver la configuración requerida, las solicitudes de la API pueden generar errores en tiempo de ejecución.

Variables de entorno del AWS SDK para Ruby

Además de las [variables de entorno entre SDK](#) compatibles con la mayoría de los AWS SDK, el AWS SDK para Ruby admite algunas variables únicas:

`AWS_SDK_CONFIG_OPT_OUT`

Si se establece la variable de entorno de AWS SDK para Ruby `AWS_SDK_CONFIG_OPT_OUT`, el archivo `config` de AWS, normalmente en `~/.aws/config`, no se usará para ningún valor de configuración.

`AMAZON_REGION`

Una variable de entorno alternativa en `AWS_REGION` para establecer la Región de AWS. Este valor solo se comprueba si no se utiliza `AWS_REGION`.

Configuración de clientes de servicio de AWS SDK para Ruby en código

Cuando la configuración se gestiona directamente en el código, el alcance de la configuración se limita a la aplicación que utiliza ese código. Dentro de esa aplicación, hay opciones para la configuración global de todos los clientes de servicio, la configuración para todos los clientes de un determinado tipo de Servicio de AWS o la configuración para una instancia de cliente de servicio específica.

`Aws.config`

Para proporcionar una configuración global en el código para todas las clases de AWS, use la [Aws.config](#) que está disponible en la gema de `aws-sdk-core`.

`Aws.config` admite dos sintaxis para usos diferentes. La configuración global se puede aplicar a todos los Servicios de AWS o a un servicio específico. Para ver la lista completa de configuraciones compatibles, consulte [Options](#) de `Client` en la Referencia de la API de AWS SDK para Ruby.

Configuración global mediante **Aws.config**

Para establecer una configuración independiente de servicios a través de `Aws.config`, utilice la sintaxis siguiente:

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

Estos ajustes se combinan en todos los clientes de servicio creados.

Ejemplo de una configuración global:

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

Si intenta usar un nombre de configuración que no es compatible a nivel mundial, se generará un error al intentar crear una instancia de un tipo de servicio que no lo admite. Si esto ocurre, use en su lugar una sintaxis específica del servicio.

Configuración específica del servicio mediante **Aws.config**

Para establecer una configuración específica del servicio a través de `Aws.config`, use la sintaxis siguiente:

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

Estos ajustes se combinan en todos los clientes de servicio creados de ese tipo de servicio.

Ejemplo de una configuración que solo se aplica a Amazon S3:

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

`<service identifier>` se puede identificar consultando el nombre del [nombre de la gema de AWS SDK para Ruby](#) correspondiente y mediante el sufijo que sigue a “aws-sdk-”. Por ejemplo:

- Para `aws-sdk-s3`, la cadena del identificador del servicio es “s3”.
- Para `aws-sdk-ecs`, la cadena del identificador del servicio es “ecs”.

Configuración de la Región de AWS para el AWS SDK para Ruby

Puede acceder a Servicios de AWS que operan en un área geográfica específica mediante Regiones de AWS. Esto puede ser útil para evitar redundancias y para que sus datos y aplicaciones se ejecuten cerca del lugar desde donde usted y los usuarios obtendrán acceso a ellos.

Important

La mayoría de los recursos residen en una Región de AWS específica y debe proporcionar la región correcta para el recurso cuando utilice el SDK.

Debe establecer una Región de AWS predeterminada para que el SDK para Ruby lo use para solicitudes de AWS. Este valor predeterminado se usa para las llamadas al método de servicio del SDK que no tiene especificadas una región.

Para obtener más información sobre la configuración de `region`, consulte [Región de AWS](#) en la Guía de referencia de las herramientas y los SDK de AWS. También incluye ejemplos sobre cómo establecer la región predeterminada mediante el archivo `config` de AWS compartido o las variables de entorno.

Orden de búsqueda de regiones para la resolución

Debe configurar una región cuando se utiliza la mayoría de los Servicios de AWS. AWS SDK para Ruby busca una región en el orden que se indica a continuación:

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`

Cómo configurar la región

En esta sección se describen diferentes maneras de configurar una región, comenzando por el enfoque más común.

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

Defina la región configurando la variable `region` en el archivo compartido `config` de AWS. Para obtener más información sobre el archivo compartido `config`, consulte [Archivos de configuración y credenciales compartidos](#) en la Guía de referencia de las herramientas y los SDK de AWS.

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

Configure la región al crear un cliente o recurso de AWS. 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 recursos de

AWS. 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')
```

Uso de AWS SDK para proveedores de credenciales de Ruby

Todas las solicitudes AWS deben estar firmadas criptográficamente con las credenciales emitidas por AWS. En tiempo de ejecución, el SDK recupera los valores de configuración para las credenciales comprobando varias ubicaciones.

La autenticación con se AWS puede gestionar fuera de su base de código. El SDK puede detectar, utilizar y actualizar automáticamente muchos métodos de autenticación mediante la cadena de proveedores de credenciales.

Para ver las opciones guiadas para empezar a AWS autenticar tu proyecto, consulta [Autenticación y acceso](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

Cadena de proveedores de credenciales

Todos SDKs tienen una serie de sitios (o fuentes) que consultan para obtener credenciales válidas y 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.

Note

Si seguiste el enfoque recomendado para los nuevos usuarios para empezar, te autenticaste mediante el inicio de sesión con las credenciales de la consola durante el [Autenticación con AWS SDK for Ruby](#) proceso. 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.

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.

La guía de referencia de AWS SDKs and Tools contiene información sobre los ajustes de configuración del SDK que utilizan todos AWS SDKs y los AWS CLI. Para obtener más información sobre cómo configurar el SDK a través del AWS config archivo compartido, consulte [Archivos de credenciales y configuración compartidos](#). Para obtener más información sobre cómo configurar el SDK mediante la configuración de variables de entorno, consulte [Compatibilidad con variables de entorno](#).

Para autenticarse AWS, el 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 SDKs y una guía de referencia de herramientas	AWS SDK para Ruby Referencia de la API
AWS claves de acceso (credenciales temporales y de larga duración)	AWS claves de acceso	Aws::Credentials Aws::SharedCredentials
token de identidad web de AWS Security Token Service (AWS STS)	Asumir el rol de proveedor de credenciales Uso de <code>role_arn</code> , <code>role_session_name</code> y <code>web_identity_token_file</code>	Aws::AssumeRoleWebIdentityCredentials
AWS IAM Identity Center. En esta guía, consulte Autenticación con AWS SDK for Ruby .	Proveedor de credenciales del IAM Identity Center	Aws::SSOCredentials
Proveedor de entidades de confianza (como <code>AWS_ROLE_ARN</code>). En esta guía, consulte Crear un token de AWS STS acceso .	Asumir el rol de proveedor de credenciales Uso de <code>role_arn</code> y <code>role_session_name</code>	Aws::AssumeRoleCredentials
Proveedor de credenciales de inicio de sesión	Proveedor de credenciales de inicio de sesión	Aws::LoginCredentials

Proveedor de credenciales por prioridad	AWS SDKs y una guía de referencia de herramientas	AWS SDK para Ruby Referencia de la API
Proveedor de credenciales de proceso	Proveedor de credenciales de proceso	Aws::ProcessCredent tials
Credenciales de Amazon Elastic Container Service (Amazon ECS)	Proveedor de credenciales de contenedor	Aws::ECSCredentials
Credenciales de perfil de instancia de Amazon Elastic Compute Cloud (Amazon EC2) (proveedor de credenciales IMDS)	Proveedor de credenciales IMDS	Aws::InstanceProfi leCredentials

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.

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. Puedes usar el [Aws::AssumeRoleCredentials](#) método para crear un token de acceso AWS Security Token Service (AWS STS).

En el siguiente ejemplo se utiliza un token de acceso para crear un objeto de cliente de Amazon S3, donde `linked::account::arn` es el nombre de 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 `role_session_name` o sobre cómo configurarlos utilizando el AWS config archivo compartido, consulte [Asumir el rol de proveedor de credenciales](#) en la Guía de referencia de AWS SDKs and Tools.

Configuración de reintentos en el AWS SDK para Ruby

El AWS SDK para Ruby proporciona un comportamiento de reintento predeterminado para las solicitudes de servicio y opciones de configuración personalizables. Las llamadas a Servicios de AWS ocasionalmente devuelven excepciones inesperadas. Pueden producirse determinados tipos errores, como la limitación o errores temporales, si se vuelve a intentar la llamada.

El comportamiento de los reintentos se puede configurar globalmente mediante variables de entorno o ajustes del archivo `config` de AWS compartido. Para obtener información sobre este enfoque, consulte [Comportamiento de los reintentos](#) en la Guía de referencia de herramientas y AWS SDK. También incluye información detallada sobre las implementaciones de las estrategias de reintento y sobre cómo elegir una u otra.

Como alternativa, estas opciones también se pueden configurar en el código, como se muestra en las siguientes secciones.

Especificación del comportamiento de reintento del cliente en el código

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) }  
)
```

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`.

Configuración de las características de observabilidad en AWS SDK para Ruby

La observabilidad es la medida en que se puede deducir el estado actual de un sistema a partir de los datos que emite. Los datos emitidos se denominan comúnmente “telemetría”. El AWS SDK para Ruby puede proporcionar los rastros como una señal de telemetría. Puede conectar un `TelemetryProvider` para recopilar y enviar datos de telemetría a un backend de observabilidad. Actualmente, el SDK es compatible con OpenTelemetry (OTel) como proveedor de telemetría y OpenTelemetry tiene muchas formas de exportar los datos de telemetría, incluido el uso de [AWS X-Ray](#) o [Amazon CloudWatch](#). Para obtener más información sobre los exportadores de OpenTelemetry para Ruby, consulte [Exportadores](#) en el sitio web de OpenTelemetry.

De forma predeterminada, el SDK no registrará ni emitirá ningún dato de telemetría. En este tema se explica cómo configurar y emitir la salida de telemetría.

La telemetría se puede configurar para un servicio específico o de forma global. El SDK para Ruby proporciona un proveedor de OpenTelemetry. También puede definir un proveedor de telemetría personalizado de su elección.

Configuración de un **OTelProvider** para un cliente de servicio

El SDK para Ruby proporciona un proveedor de OpenTelemetry llamado [OTelProvider](#). El siguiente ejemplo configura la exportación de telemetría mediante OpenTelemetry para el cliente del servicio Amazon Simple Storage Service. En este sencillo ejemplo, la variable de entorno `OTEL_TRACES_EXPORTER` de OpenTelemetry se utiliza para exportar los rastros a la salida de la consola cuando se ejecuta el código. Para obtener más información sobre `OTEL_TRACES_EXPORTER`, consulte [Selección de exportadores](#) en la documentación de OpenTelemetry.

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

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

OpenTelemetry::SDK.configure

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

```
client.list_buckets
```

El ejemplo de código anterior muestra los pasos para configurar la salida de rastreo para un cliente de servicio:

1. Requiere dependencias de OpenTelemetry.
 - a. [opentelemetry-sdk](#) para usar `Aws::Telemetry::OTelProvider`.
 - b. [opentelemetry-exporter-otlp](#) para exportar datos de telemetría.
2. Llame a `OpenTelemetry::SDK.configure` para configurar el OpenTelemetry SDK con sus valores de configuración predeterminados.
3. Con el SDK del proveedor de OpenTelemetry de Ruby, se crea una instancia del `OTelProvider` para pasarla como opción de configuración al cliente de servicio que desea rastrear.

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

Con estos pasos, cualquier método que se invoque en ese cliente de servicio emitirá datos de rastreo.

A continuación, se muestra un ejemplo del resultado de rastreo generado por la llamada al método `list_buckets` de Amazon S3:

Ejemplo de salida de rastreo de OpenTelemetry

```
#<struct OpenTelemetry::SDK::Trace::SpanData
  name="Handler.NetHttp",
  kind=:internal,
  status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
  parent_span_id="\xBFb\xC9\xFD\xA6F!\xE1",
  total_recorded_attributes=7,
  total_recorded_events=0,
  total_recorded_links=0,
  start_timestamp=1736190567061767000,
  end_timestamp=1736190567317160000,
  attributes=
  {"http.method"=>"GET",
   "net.protocol.name"=>"http",
   "net.protocol.version"=>"1.1",
   "net.peer.name"=>"s3.amazonaws.com",
```

```

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

```

```

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

```

La salida de rastreo anterior tiene dos intervalos de datos. Cada entrada de rastreo proporciona metadatos adicionales sobre el evento en uno o más atributos.

Configuración de un **OTelProvider** para todos los clientes de servicio

En lugar de activar la telemetría para un cliente de servicio específico, como se ha explicado en la sección anterior, tiene la opción de activar la telemetría de forma global.

Para emitir datos de telemetría para todos los clientes de servicio de AWS, puede configurar el proveedor de telemetría en `Aws.config` antes de crear clientes de servicio.

```

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

```

Con esta configuración, cualquier cliente de servicio que se cree posteriormente emitirá automáticamente la telemetría. Para obtener más información sobre el uso de `Aws.config` para configurar ajustes global, consulte [Aws.config](#).

Configuración de un proveedor de telemetría personalizado

Si no quiere usar OpenTelemetry como proveedor de telemetría, el AWS SDK para Ruby le permite implementar un proveedor personalizado. Podría ser útil usar como ejemplo la [implementación de OTelProvider](#) que está disponible en el repositorio de GitHub de AWS SDK para Ruby. Para obtener más información sobre el contexto, consulte las notas de [Module: Aws::Telemetry](#) en la Referencia de la API de AWS SDK para Ruby.

Atributos del intervalo

Los rastros son el resultado de la telemetría. Un conjunto de rastros consta de uno o varios intervalos. Los rastros tienen atributos que incluyen metadatos adicionales que se incluyen automáticamente cuando es apropiado para la llamada del método. A continuación, se muestra una lista de los atributos compatibles con el SDK para Ruby, donde:

- Nombre del atributo: el nombre que se usa para etiquetar los datos que aparecen en el rastro.
- Tipo: el tipo de datos del valor.
- Descripción: una descripción de lo que representa el valor.

Attribute Name	Type	Description
<code>error</code>	Boolean	True if the unit of work was unsuccessful. Otherwise, false.
<code>exception.message</code>	String	The exception or error message.
<code>exception.stacktrace</code>	String	A stacktrace as provided by the language runtime if available.
<code>exception.type</code>	String	The type (fully qualified name) of the exception or error.
<code>rpc.system</code>	String	The remote system identifier set to 'aws-api'.
<code>rpc.method</code>	String	The name of the operation being invoked.
<code>rpc.service</code>	String	The name of the remote service.
<code>aws.request_id</code>	String	The AWS request ID returned in the response headers, per

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

 Tip

OpenTelemetry-Ruby tiene implementaciones adicionales que se integran con el SDK para el soporte de telemetría actual de Ruby. Para obtener más información, consulte [OpenTelemetry AWS-SDK Instrumentation](#) en el repositorio de open-telemetry GitHub.

Configuración de los ajustes por HTTP en el AWS SDK para Ruby

Configuración de un punto de conexión no estándar

La región se utiliza para crear un punto de conexión SSL que se utilizará en las solicitudes de AWS. 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 utilizar un punto de conexión de su elección para las solicitudes de la API y para hacer que esa opción persista, consulte la opción de configuración de [puntos de conexión específicos del servicio](#) de la Guía de referencia de las herramientas y los SDK de AWS.

Uso de AWS SDK para Ruby

Esta sección contiene información sobre el desarrollo de software con AWS SDK para Ruby; por ejemplo, acerca de cómo utilizar algunas de las características avanzadas del SDK.

La [Guía de referencia de las herramientas y los SDK de AWS](#) también contiene configuraciones, características y otros conceptos fundamentales comunes a muchos de los SDK de AWS.

Temas

- [Realización de solicitudes de Servicio de AWS con el AWS SDK para Ruby](#)
- [Uso de la utilidad REPL de AWS SDK para Ruby](#)
- [Uso de AWS SDK para Ruby con Ruby on Rails](#)
- [Depuración mediante información de rastreo de red de un cliente AWS SDK para Ruby](#)
- [Agregación de pruebas con stubbing a la aplicación AWS SDK para Ruby](#)
- [Uso de resultados paginados en el AWS SDK para Ruby](#)
- [Uso de esperadores en AWS SDK para Ruby](#)

Realización de solicitudes de Servicio de AWS con el AWS SDK para Ruby

Para acceder a los Servicios de AWS mediante programación, los SDK utilizan una clase de cliente para cada Servicio de AWS. Por ejemplo, si su aplicación necesita acceder a Amazon EC2, su aplicación crearía un objeto de cliente de Amazon EC2 para interactuar con ese servicio. A continuación, utiliza el cliente de servicio para realizar solicitudes al mismo Servicio de AWS.

Para realizar una solicitud a un Servicio de AWS, primero debe crear y [configurar](#) un cliente de servicio. Para cada Servicio de AWS que utilice el código, tiene su propia gema y su propio tipo específico para interactuar con él. El cliente expone un método para cada operación de la API expuesta por el servicio.

Cada cliente de servicio requiere una Región de AWS y un proveedor de credenciales. El SDK usa estos valores para enviar solicitudes a la región correcta de los recursos y para firmar las solicitudes con las credenciales correctas. Puede especificar estos valores mediante programación en el código o hacer que se carguen automáticamente desde el entorno.

- Al crear una instancia de una clase de cliente, se deben proporcionar las credenciales de AWS. Para ver el orden en el que el SDK comprueba los proveedores de autenticación, consulte [Cadena de proveedores de credenciales](#).
- El SDK tiene una serie de lugares (u orígenes) que comprueba para encontrar un valor para la configuración global. Para obtener más información, consulte [Prioridad de los ajustes](#).

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 y sigue la convención `Aws::<service identifier>::Client`. Por ejemplo, [Aws::S3::Client](#) proporciona una interfaz para el servicio Amazon Simple Storage Service y [Aws::SQS::Client](#) proporciona una interfaz para el servicio Amazon Simple Queue Service.

Todas las clases de clientes para todos los Servicios de AWS son seguros para subprocessos.

Puede pasar las opciones de configuración directamente a los constructores de clientes y recursos. Estas opciones tienen prioridad sobre el entorno y los valores predeterminados de `Aws.config`.

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

Uso de 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

- [Instalación de 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 `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.

Uso de AWS SDK para Ruby 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 gema `aws-sdk-rails` para que la integración con Rails sea sencilla. Puede utilizar AWS Elastic Beanstalk, AWS OpsWorks, AWS CodeDeploy o [AWS Rails Provisioner](#) para implementar y ejecutar aplicaciones de Rails en la nube de AWS.

Para obtener información sobre cómo instalar y usar la gema `aws-sdk-rails`, consulte el repositorio de GitHub <https://github.com/aws/aws-sdk-rails>.

Depuración mediante información de rastreo de red de un cliente AWS SDK para Ruby

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/
iAX5E30XRBzJ37cfc8T4D7ELC1KFELM=\r\n"
-> "x-amz-request-id: 5MD4APQQS815QVBR\r\n"
-> "Date: Thu, 27 Apr 2023 14:31:47 GMT\r\n"
-> "x-amz-bucket-region: us-east-1\r\n"
-> "x-amz-access-point-alias: false\r\n"
-> "Content-Type: application/xml\r\n"
-> "Server: AmazonS3\r\n"
-> "\r\n"
Conn keep-alive
Bucket bucket_name exists
```

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](#).

Agregación de pruebas con stubbing a la aplicación AWS SDK para Ruby

Obtenga información sobre cómo dissociar 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
```

Uso de resultados paginados en el AWS SDK para Ruby

Muchas operaciones de AWS devuelven resultados truncados cuando la carga útil es demasiado grande para que se devuelva en una sola respuesta. En su lugar, el servicio devuelve una parte de los datos y un token para recuperar el siguiente conjunto de elementos. Este patrón se conoce como paginación.

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.

Uso de esperadores en AWS SDK para Ruby

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

Se ha producido un estado de error durante la espera.

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

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

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

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

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

Se ha producido un error inesperado durante la espera.

[Aws::Writers::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::Writers::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
```

Ejemplos de código de SDK para Ruby

Los ejemplos de código de este tema muestran cómo usar el AWS SDK para 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 distintas funciones de servicio, 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.

Algunos servicios contienen categorías de ejemplo adicionales que muestran cómo aprovechar las bibliotecas o funciones específicas del servicio.

Services

- [Ejemplos de Aurora con SDK para Ruby](#)
- [Ejemplos de Auto Scaling con SDK para Ruby](#)
- [CloudTrail ejemplos de uso de SDK for Ruby](#)
- [CloudWatch ejemplos de uso de SDK for Ruby](#)
- [Ejemplos del proveedor de identidad de Amazon Cognito con SDK para Ruby](#)
- [Ejemplos de Amazon Comprehend con SDK para Ruby](#)
- [Ejemplos de Amazon DocumentDB con SDK para Ruby](#)
- [Ejemplos de DynamoDB usando SDK para Ruby](#)
- [EC2 Ejemplos de Amazon que utilizan SDK for Ruby](#)
- [Ejemplos de Elastic Beanstalk con SDK para Ruby](#)
- [EventBridge ejemplos de uso de SDK for Ruby](#)
- [AWS Glue ejemplos de uso de SDK for Ruby](#)
- [Ejemplos de IAM usando SDK para Ruby](#)
- [Ejemplos de Kinesis con SDK para Ruby](#)
- [AWS KMS ejemplos de uso de SDK for Ruby](#)
- [Ejemplos de Lambda usando SDK para Ruby](#)

- [Ejemplos de Amazon MSK con SDK para Ruby](#)
- [Ejemplos de Amazon Polly con SDK para Ruby](#)
- [Ejemplos de Amazon RDS usando SDK para Ruby](#)
- [Ejemplos de Amazon S3 usando SDK para Ruby](#)
- [Ejemplos de Amazon SES con SDK para Ruby](#)
- [Ejemplos de Amazon SES API v2 con SDK para Ruby](#)
- [Ejemplos de Amazon SNS usando SDK para Ruby](#)
- [Ejemplos de Amazon SQS usando SDK para Ruby](#)
- [AWS STS ejemplos de uso de SDK for Ruby](#)
- [Ejemplos de Amazon Textract con SDK para Ruby](#)
- [Ejemplos de Amazon Translate con SDK para Ruby](#)

Ejemplos de Aurora con SDK para Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)

Introducción

Introducción a Aurora

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Describir DBClusters](#) en la referencia de la AWS SDK para Ruby API.

Ejemplos de Auto Scaling con SDK para Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)

Introducción

Introducción al escalado automático

El siguiente ejemplo de código muestra cómo empezar a usar Auto Scaling.

SDK para Ruby

Note

Hay más información al respecto 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}/\n#{group.desired_capacity}")
      end
    end
  end
end
```

```
        @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 obtener más información sobre la API, consulta [DescribeAutoScalingGroups](#) la Referencia AWS SDK para Ruby de la API.

CloudTrail ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

CreateTrail

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

SDK para Ruby

 Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateTrail](#) la Referencia AWS SDK para Ruby de la API.

DeleteTrail

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

SDK para Ruby

Note

Hay más información al respecto 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 => e
    puts "Got error trying to delete trail: #{trail_name}:"
    puts e
    exit 1
end
```

- Para obtener más información sobre la API, consulta [DeleteTrail](#) en la Referencia AWS SDK para Ruby de la API.

ListTrails

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

SDK para Ruby

Note

Hay más información al respecto en 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
end
```

- Para obtener más información sobre la API, consulta [ListTrails](#) la Referencia AWS SDK para Ruby de la API.

LookupEvents

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [LookupEvents](#) la Referencia AWS SDK para Ruby de la API.

CloudWatch ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

DescribeAlarms

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [DescribeAlarms](#) la Referencia AWS SDK para Ruby de la API.

DescribeAlarmsForMetric

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

SDK para Ruby

Note

Hay más información al respecto 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}"
    end
  end
end

```

```
puts "State reason:    #{alarm.state_reason}"
puts "Metric:         #{alarm.metric_name}"
puts "Namespace:      #{alarm.namespace}"
puts "Statistic:       #{alarm.statistic}"
puts "Period:         #{alarm.period}"
puts "Unit:           #{alarm.unit}"
puts "Eval. periods:   #{alarm.evaluation_periods}"
puts "Threshold:       #{alarm.threshold}"
puts "Comp. operator:  #{alarm.comparison_operator}"

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

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

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

puts 'Dimensions:'
if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
  alarm.dimensions.each do |d|
    puts "  Name: #{d.name}, Value: #{d.value}"
  end
else
  puts '  None for this alarm.'
end
end
else
  puts 'No alarms found.'
end
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 obtener más información sobre la API, consulta [DescribeAlarmsForMetric](#) la Referencia AWS SDK para Ruby de la API.

DisableAlarmActions

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

SDK para Ruby

 Note

Hay más información al respecto 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])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
```

```

dimensions = [
  {
    name: "BucketName",
    value: "amzn-s3-demo-bucket"
  },
  {
    name: 'StorageType',
    value: 'AllStorageTypes'
  }
]
period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
unit = 'Count'
evaluation_periods = 1 # More than one day.
threshold = 1 # One object.
comparison_operator = 'GreaterThanThreshold' # More than one object.
# Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
region = 'us-east-1'

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

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

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

```

```

    end
  end

  run_me if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [DisableAlarmActions](#) la Referencia AWS SDK para Ruby de la API.

ListMetrics

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

SDK para Ruby

Note

Hay más información al respecto 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
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 obtener más información sobre la API, consulta [ListMetrics](#) la Referencia AWS SDK para Ruby de la API.

PutMetricAlarm

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

SDK para Ruby

Note

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

```

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

- Para obtener más información sobre la API, consulta [PutMetricAlarm](#) la Referencia AWS SDK para Ruby de la API.

PutMetricData

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

SDK para Ruby

 Note

Hay más información al respecto 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}'."
true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
      "'#{metric_namespace}': #{e.message}"
  false
end

```

- Para obtener más información sobre la API, consulta [PutMetricData](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos del proveedor de identidad de Amazon Cognito con SDK para Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)

Introducción

Introducción a Amazon Cognito

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

SDK para Ruby

Note

Hay más información al respecto. 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}")
      end
    end
  end
end
```

```
        @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 obtener más información sobre la API, consulta [ListUserPools](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Amazon Comprehend con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el 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.

SDK para 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 con SDK para Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología 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 desencadenado al recibir registros de una secuencia de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDK para Ruby

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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 SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS SDK para 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 distintas funciones de servicio, 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a DynamoDB

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

SDK para Ruby

Note

Hay más información al respecto. 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 obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK para Ruby de la API.

Conceptos básicos

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.
- Escribir los datos de películas en la tabla a partir de un archivo JSON de ejemplo.
- 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.

SDK para Ruby

Note

Hay más información al respecto 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
    ]
  )
end
```

```

    ],
    attribute_definitions: [
      { attribute_name: 'year', attribute_type: 'N' },
      { attribute_name: 'title', attribute_type: 'S' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end

```

Crear una función auxiliar para descargar y extraer el archivo JSON 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 unless continue.eql?('y')
  end
end
print "\nDone!\n".green

new_step(6, 'Scan for a batch of items using a filter expression.')
years = {}
years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
2018:')
years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
releases = dynamodb_wrapper.scan_items(years)
if !releases.empty?
  puts("Found #{releases.length} movies.")
  count = Question.ask(
    'How many do you want to see? ', method(:is_int), in_range(1, releases.length)
  )
  puts("Here are your #{count} movies:")
  releases.take(count).each do |release|

```

```

        puts("\t#{release['title']}")
      end
    else
      puts("I don't know about any movies released between #{years[:start]} "\
        "and #{years[:end]}".")
    end
  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 obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Ruby .
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)

- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

Acciones

BatchExecuteStatement

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

SDK para 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, :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
  end
end
```

```

    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
end

```

Eliminar un lote de elementos con PartiQL.

```

class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

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

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

```

- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para Ruby de la API.

BatchWriteItem

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

SDK para Ruby

 Note

Hay más información al respecto 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, :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}")
    end
  end
end
```

```
    raise
  end
```

- Para obtener más información sobre la API, consulta [BatchWriteItem](#) la Referencia AWS SDK para Ruby de la API.

CreateTable

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

SDK para Ruby

Note

Hay más información al respecto 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, :table_name, :table

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

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
```

```

    key_schema: [
      { attribute_name: 'year', key_type: 'HASH' }, # Partition key
      { attribute_name: 'title', key_type: 'RANGE' } # Sort key
    ],
    attribute_definitions: [
      { attribute_name: 'year', attribute_type: 'N' },
      { attribute_name: 'title', attribute_type: 'S' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end

```

- Para obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK para Ruby de la API.

DeleteItem

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

SDK para Ruby

Note

Hay más información al respecto 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, :table

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

```

```
# 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 obtener más información sobre la API, consulta [DeleteItem](#) la Referencia AWS SDK para Ruby de la API.

DeleteTable

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

SDK para Ruby

Note

Hay más información al respecto 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, :table_name, :table

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

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

- Para obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK para Ruby de la API.

DescribeTable

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

SDK para Ruby

Note

Hay más información al respecto 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, :table_name, :table

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

  # Determines whether a table exists. As a side effect, stores the table in
```

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

- Para obtener más información sobre la API, consulta [DescribeTable](#) la Referencia AWS SDK para Ruby de la API.

ExecuteStatement

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

SDK para Ruby

Note

Hay más información al respecto 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, :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, :table

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

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

```

Añadir un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

Eliminar un solo elemento con PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
```

```
# @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
def delete_item_by_title(title, year)
  request = {
    statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
    parameters: [title, year]
  }
  @dynamodb.client.execute_statement(request)
end
```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para Ruby de la API.

GetItem

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

SDK para Ruby

Note

Hay más información al respecto 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, :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 })
  end
end
```

```
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 obtener más información sobre la API, consulta [GetItem](#) la Referencia AWS SDK para Ruby de la API.

ListTables

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

SDK para Ruby

Note

Hay más información al respecto 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, :table_name, :table

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

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

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

- Para obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK para Ruby de la API.

PutItem

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

SDK para Ruby

Note

Hay más información al respecto 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, :table

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

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

```

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

```

- Para obtener más información sobre la API, consulta [PutItem](#) la Referencia AWS SDK para Ruby de la API.

Query

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

SDK para Ruby

Note

Hay más información al respecto 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, :table

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

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.

```

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

- Para obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK para Ruby .

Scan

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

SDK para 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, :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
end

```

- Para obtener información sobre la API, consulte [Scan](#) en la referencia de la API de AWS SDK para Ruby .

UpdateItem

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

SDK para 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, :table

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

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

- Para obtener más información sobre la API, consulta [UpdateItem](#) la Referencia AWS SDK para Ruby de la API.

Escenarios

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un lote de elementos mediante la ejecución de varias instrucciones SELECT.
- Agregar un lote de elementos mediante la ejecución de varias instrucciones INSERT.
- Actualizar un lote de elementos con la ejecución de varias instrucciones UPDATE.
- Eliminación de un lote de elementos con la ejecución de varias instrucciones DELETE.

SDK para Ruby

Note

Hay más información al respecto 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.')
return unless scaffold.exists?(table_name)

scaffold.delete_table
end
```

- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para Ruby de la API.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtener un artículo mediante una instrucción SELECT.
- Agregar un elemento mediante una instrucción INSERT.
- Actualizar un elemento mediante una instrucción UPDATE.
- Eliminar un elemento mediante una instrucción DELETE.

SDK para Ruby

Note

Hay más información al respecto 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.to_s.yellow
print "\n\nDone!\n".green

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

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

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

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

scaffold.delete_table
end
```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de DynamoDB

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

SDK para Ruby

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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']
end
```

```
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 parcial por lotes para las funciones de Lambda que reciben eventos de una secuencia 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.

SDK para Ruby

Note

Hay más información [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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
```

EC2 Ejemplos de Amazon que utilizan SDK for Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK para Ruby con Amazon EC2.

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)

Introducción

Hola Amazon EC2

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon EC2.

SDK para Ruby

Note

Hay más información al respecto 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}")
  end
end
```

```

        @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
end

```

- Para obtener más información sobre la API, consulta [DescribeSecurityGroups](#) la Referencia AWS SDK para Ruby de la API.

Acciones

AllocateAddress

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

SDK para Ruby

Note

Hay más información al respecto 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')
  response.allocation_id
rescue StandardError => e

```

```
puts "Error allocating Elastic IP address: #{e.message}"
'Error'
end
```

- Para obtener más información sobre la API, consulta [AllocateAddress](#) la Referencia AWS SDK para Ruby de la API.

AssociateAddress

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

SDK para Ruby

Note

Hay más información al respecto 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 association ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
```

```
allocation_id,  
instance_id  
)  
response = ec2_client.associate_address(  
  allocation_id: allocation_id,  
  instance_id: instance_id  
)  
response.association_id  
rescue StandardError => e  
  puts "Error associating Elastic IP address with instance: #{e.message}"  
  'Error'  
end
```

- Para obtener más información sobre la API, consulta [AssociateAddress](#) la Referencia AWS SDK para Ruby de la API.

CreateKeyPair

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

SDK para Ruby

Note

Hay más información al respecto 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}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

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

```

```

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

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

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

```

```

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [CreateKeyPair](#) la Referencia AWS SDK para Ruby de la API.

CreateRouteTable

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

SDK para Ruby

Note

Hay más información al respecto 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}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''

```

```

# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
    'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
    'TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
    'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
    "'0.0.0.0/0' my-key my-value us-west-2"
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-0b6f769731EXAMPLE'
  subnet_id = 'subnet-03d9303b57EXAMPLE'
  gateway_id = 'igw-06ca90c011EXAMPLE'
  destination_cidr_block = '0.0.0.0/0'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  subnet_id = ARGV[1]
  gateway_id = ARGV[2]
  destination_cidr_block = ARGV[3]
  tag_key = ARGV[4]
  tag_value = ARGV[5]
  region = ARGV[6]
end

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

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts 'Route table created and associated.'
else

```

```

    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [CreateRouteTable](#) la Referencia AWS SDK para Ruby de la API.

CreateSecurityGroup

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

SDK para Ruby

Note

Hay más información al respecto 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}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

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

```

```

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

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

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

```

```

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

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

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

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

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

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

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

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

```

```

    end
  end

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

    return if sg.ip_permissions_egress.empty?

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

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

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

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

    if security_group_exists
      add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
    from_port_http, to_port_http, cidr_ip_range_http)
    end
  end

```

```

    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh, from_port_ssh,
to_port_ssh, cidr_ip_range_ssh)
  end

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

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

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

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

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

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

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

```

```

end

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

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

run_me if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [CreateSecurityGroup](#) la Referencia AWS SDK para Ruby de la API.

CreateSubnet

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

SDK para Ruby

Note

Hay más información al respecto 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}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

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

```

```
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

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

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta [CreateSubnet](#) la Referencia AWS SDK para Ruby de la API.

CreateVpc

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

SDK para Ruby

Note

Hay más información al respecto 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}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''

```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta [CreateVpc](#) Referencia AWS SDK para Ruby de la API.

DescribeInstances

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

SDK para Ruby

Note

Hay más información al respecto 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
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 obtener más información sobre la API, consulta [DescribeInstances](#) la Referencia AWS SDK para Ruby de la API.

DescribeRegions

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

SDK para Ruby

Note

Hay más información al respecto 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.
end
```

```

max_region_string_length = 16
max_endpoint_string_length = 33
# Print header.
print 'Region'
print ' ' * (max_region_string_length - 'Region'.length)
print "  Endpoint\n"
print '-' * max_region_string_length
print ' '
print '-' * max_endpoint_string_length
print "\n"
# Print Regions and their endpoints.
result.regions.each do |region|
  print region.region_name
  print ' ' * (max_region_string_length - region.region_name.length)
  print ' '
  print region.endpoint
  print "\n"
end
end

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

```

```

print "\n"
# Print Regions, Availability Zones, and their states.
result.availability_zones.each do |zone|
  print zone.region_name
  print ' ' * (max_region_string_length - zone.region_name.length)
  print ' '
  print zone.zone_name
  print ' ' * (max_zone_string_length - zone.zone_name.length)
  print ' '
  print zone.state
  # Print any messages for this Availability Zone.
  if zone.messages.count.positive?
    print "\n"
    puts '  Messages for this zone:'
    zone.messages.each do |message|
      print "    #{message.message}\n"
    end
  end
  print "\n"
end
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 obtener más información sobre la API, consulta [DescribeRegions](#) la Referencia AWS SDK para Ruby de la API.

ReleaseAddress

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

SDK para Ruby

Note

Hay más información al respecto 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)
    true
  rescue StandardError => e
    puts("Error releasing Elastic IP address: #{e.message}")
    false
  end
end

```

- Para obtener más información sobre la API, consulta [ReleaseAddress](#) la Referencia AWS SDK para Ruby de la API.

StartInstances

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

SDK para Ruby

Note

Hay más información al respecto 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.'
true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

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

```
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

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

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

puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta [StartInstances](#) la Referencia AWS SDK para Ruby de la API.

StopInstances

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

SDK para Ruby

Note

Hay más información al respecto 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
end

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

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.

```

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

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

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

puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información sobre la API, consulta [StopInstances](#) la Referencia AWS SDK para Ruby de la API.

TerminateInstances

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

SDK para Ruby

Note

Hay más información al respecto 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.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.

```

```

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

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

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

puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [TerminateInstances](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Elastic Beanstalk con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

DescribeApplications

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

SDK para Ruby

Note

Hay más información al respecto. 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
end

```

- Para obtener más información sobre la API, consulta [DescribeApplications](#) la Referencia AWS SDK para Ruby de la API.

ListAvailableSolutionStacks

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

SDK para Ruby

Note

Hay más información al respecto 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
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 obtener más información sobre la API, consulta [ListAvailableSolutionStacks](#) la Referencia AWS SDK para Ruby de la API.

UpdateApplication

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

SDK para Ruby

 Note

Hay más información al respecto 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
end
```

```

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

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

# Creates a new application version and deploys it
def create_and_deploy_new_application_version(zip_file_name)
  version_label = File.basename(zip_file_name, '.zip')
  @eb_client.create_application_version(
    process: false,
    application_name: @app_name,
    version_label: version_label,
    source_bundle: {
      s3_bucket: fetch_bucket_name,
      s3_key: "#{@app_name}/#{zip_file_name}"
    },
    description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
  )
  update_environment(version_label)
rescue Aws::Errors::ServiceError => e
  @logger.error("Failed to create or deploy application version: #{e.message}")
end

# Updates the environment to the new application version
def update_environment(version_label)
  env_name = fetch_environment_name
  @eb_client.update_environment(
    environment_name: env_name,

```

```
        version_label: version_label
      )
    rescue Aws::Errors::ServiceError => e
      @logger.error("Failed to update environment: #{e.message}")
    end

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

- Para obtener más información sobre la API, consulta [UpdateApplication](#) la Referencia AWS SDK para Ruby de la API.

EventBridge ejemplos de uso de SDK for Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Escenarios](#)

Escenarios

Crear y activar una regla

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

SDK para Ruby

 Note

Hay más información al respecto 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'
```

Comprobar si el tema de Amazon Simple Notification Service (Amazon SNS) especificado existe entre los que se proporcionan 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
end
```

```

    false
  end

```

Comprobar si el tema especificado existe entre los disponibles para el intermediario en Amazon SNS.

```

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

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

```

Crear un tema en Amazon SNS y después suscribe una dirección de correo electrónico para recibir notificaciones sobre dicho 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.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Compruebe si el rol especificado AWS Identity and Access Management (IAM) existe entre los que se proporcionan 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
  false
end
```

Comprobar si el rol especificado existe entre los disponibles para el intermediario en 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
  end
  while response.next_page?
```

```

    response = response.next_page
    next unless response.roles.count.positive?

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

```

Crear un rol en IAM.

```

# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'

```

```

        },
        'Action': 'sts:AssumeRole'
      }
    ]
  }.to_json,
  path: '/',
  role_name: role_name
)
puts "Role created with ARN '#{response.role.arn}'."
puts 'Adding access policy to role...'
iam_client.put_role_policy(
  policy_document: {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Sid': 'CloudWatchEventsFullAccess',
        'Effect': 'Allow',
        'Resource': '*',
        'Action': 'events:*'
      },
      {
        'Sid': 'IAMPassRoleForCloudWatchEvents',
        'Effect': 'Allow',
        'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
        'Action': 'iam:PassRole'
      }
    ]
  }.to_json,
  policy_name: 'CloudWatchEventsPolicy',
  role_name: role_name
)
puts 'Access policy added to role.'
response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  'Error'
end

```

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
  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
  end
end
```

```

    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

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

```

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.positive?
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end

```

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
  )
end

```

```

)
if response.log_groups.count.positive?
  response.log_groups.each do |log_group|
    if log_group.log_group_name == log_group_name
      puts 'Log group found.'
      return true
    end
  end
end
puts 'Log group not found.'
false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end

```

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.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  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
      }
    ]
  }
end

```

```

    }
  ]
}
event[:sequence_token] = sequence_token unless sequence_token.empty?

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

```

Reinicie una instancia de Amazon Elastic Compute Cloud (Amazon EC2) 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
  )

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

```

```

    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end

```

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
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

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Ruby .
 - [PutEvents](#)
 - [PutRule](#)

AWS Glue ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)

Introducción

Hola AWS Glue

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

SDK para 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 obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK para Ruby de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rastreador que rastree un bucket de Amazon S3 público y generar una base de datos de metadatos con formato CSV.
- Enumera información sobre bases de datos y tablas en tu AWS Glue Data Catalog.
- Crear un trabajo para extraer datos CSV del bucket de S3, transformar los datos y cargar el resultado con formato JSON en otro bucket 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](#).

SDK para Ruby

Note

Hay más información al respecto 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
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
  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)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
  end
end

```

```

    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
end

private

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

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

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

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

```

```

end

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

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

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

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

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

  puts 'Demo complete.'
end

```

Cree un script ETL 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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Ruby .

- [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 utilizar CreateCrawler.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateCrawler](#) la Referencia AWS SDK para Ruby de la API.

CreateJob

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

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información sobre la API, consulta [CreateJob](#) la Referencia AWS SDK para Ruby de la API.

DeleteCrawler

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [DeleteCrawler](#) la Referencia AWS SDK para Ruby de la API.

DeleteDatabase

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [DeleteDatabase](#) la Referencia AWS SDK para Ruby de la API.

DeleteJob

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [DeleteJob](#) la Referencia AWS SDK para Ruby de la API.

DeleteTable

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK para Ruby de la API.

GetCrawler

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

SDK para Ruby

 Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [GetCrawler](#) la Referencia AWS SDK para Ruby de la API.

GetDatabase

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [GetDatabase](#) la Referencia AWS SDK para Ruby de la API.

GetJobRun

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [GetJobRun](#) la Referencia AWS SDK para Ruby de la API.

GetJobRuns

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [GetJobRuns](#) la Referencia AWS SDK para Ruby de la API.

GetTables

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

- Para obtener más información sobre la API, consulta [GetTables](#) la Referencia AWS SDK para Ruby de la API.

ListJobs

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
end
```

- Para obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK para Ruby de la API.

StartCrawler

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
end
```

- Para obtener más información sobre la API, consulta [StartCrawler](#) la Referencia AWS SDK para Ruby de la API.

StartJobRun

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [StartJobRun](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de IAM usando SDK para Ruby

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

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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)

Introducción

Introducción a IAM

El siguiente ejemplo de código muestra cómo empezar a utilizar IAM.

SDK para Ruby

 Note

Hay más información al respecto. 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 obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK para Ruby de la API.

Conceptos básicos

Conceptos básicos

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 los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado 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.

SDK para Ruby

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un usuario de IAM y un rol que conceda permiso para enumerar los 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
  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
      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
    end
  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}.")
end

```

```

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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Ruby .

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

- [PutUserPolicy](#)

Acciones

AttachRolePolicy

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

SDK para 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
  end
end
```

```

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
end

```

- Para obtener más información sobre la API, consulta [AttachRolePolicy](#) la Referencia AWS SDK para Ruby de la API.

AttachUserPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [AttachUserPolicy](#) la Referencia AWS SDK para Ruby de la API.

CreateAccessKey

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

SDK para Ruby

Note

Hay más información al respecto 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
  @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
  []
rescue StandardError => e
  @logger.error("Error listing access keys: #{e.message}")
  []
end

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

# Deactivates an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
# @return [Boolean]
def deactivate_access_key(user_name, access_key_id)
  @iam_client.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,

```

```

        status: 'Inactive'
      )
      true
    rescue StandardError => e
      @logger.error("Error deactivating access key: #{e.message}")
      false
    end

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

```

- Para obtener más información sobre la API, consulta [CreateAccessKey](#) la Referencia AWS SDK para Ruby de la API.

CreateAccountAlias

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
end
```

```

rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end

```

- Para obtener más información sobre la API, consulta [CreateAccountAlias](#) la Referencia AWS SDK para Ruby de la API.

CreatePolicy

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

SDK para Ruby

Note

Hay más información al respecto 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
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 obtener más información sobre la API, consulta [CreatePolicy](#) la Referencia AWS SDK para Ruby de la API.

CreateRole

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

SDK para Ruby

 Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateRole](#) la Referencia AWS SDK para Ruby de la API.

CreateServiceLinkedRole

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateServiceLinkedRole](#) la Referencia AWS SDK para Ruby de la API.

CreateUser

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateUser](#) la Referencia AWS SDK para Ruby de la API.

DeleteAccessKey

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

SDK para Ruby

 Note

Hay más información al respecto 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
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

```
@logger.info("Access key created for user '#{user_name}':  
#{access_key.access_key_id}")  
  access_key  
rescue Aws::IAM::Errors::LimitExceeded  
  @logger.error('Error creating access key: limit exceeded. Cannot create more.')  
  nil  
rescue StandardError => e  
  @logger.error("Error creating access key: #{e.message}")  
  nil  
end  
  
# Deactivates an access key  
#  
# @param user_name [String] The name of the user.  
# @param access_key_id [String] The ID for the access key.  
# @return [Boolean]  
def deactivate_access_key(user_name, access_key_id)  
  @iam_client.update_access_key(  
    user_name: user_name,  
    access_key_id: access_key_id,  
    status: 'Inactive'  
  )  
  true  
rescue StandardError => e  
  @logger.error("Error deactivating access key: #{e.message}")  
  false  
end  
  
# Deletes an access key  
#  
# @param user_name [String] The name of the user.  
# @param access_key_id [String] The ID for the access key.  
# @return [Boolean]  
def delete_access_key(user_name, access_key_id)  
  @iam_client.delete_access_key(  
    user_name: user_name,  
    access_key_id: access_key_id  
  )  
  true  
rescue StandardError => e  
  @logger.error("Error deleting access key: #{e.message}")  
  false  
end  
end
```

- Para obtener más información sobre la API, consulta [DeleteAccessKey](#) la Referencia AWS SDK para Ruby de la API.

DeleteAccountAlias

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

```
end

# Creates an AWS account alias.
#
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
def create_account_alias(account_alias)
  @iam_client.create_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating account alias: #{e.message}")
  false
end

# Deletes an AWS account alias.
#
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
def delete_account_alias(account_alias)
  @iam_client.delete_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end
```

- Para obtener más información sobre la API, consulta [DeleteAccountAlias](#) la Referencia AWS SDK para Ruby de la API.

DeleteRole

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

SDK para Ruby

Note

Hay más información al respecto 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)
  # 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 obtener más información sobre la API, consulta [DeleteRole](#) la Referencia AWS SDK para Ruby de la API.

DeleteServerCertificate

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

SDK para Ruby

 Note

Hay más información al respecto 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
  end
end
```

```

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

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

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

```

- Para obtener más información sobre la API, consulta [DeleteServerCertificate](#) la Referencia AWS SDK para Ruby de la API.

DeleteServiceLinkedRole

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

SDK para Ruby

 Note

Hay más información al respecto 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)
  return if e.code == 'NoSuchEntity'

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

- Para obtener más información sobre la API, consulta [DeleteServiceLinkedRole](#) la Referencia AWS SDK para Ruby de la API.

DeleteUser

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK para Ruby de la API.

DeleteUserPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [DeleteUserPolicy](#) la Referencia AWS SDK para Ruby de la API.

DetachRolePolicy

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

```

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

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

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

```

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

- Para obtener más información sobre la API, consulta [DetachRolePolicy](#) la Referencia AWS SDK para Ruby de la API.

DetachUserPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información sobre la API, consulta [DetachUserPolicy](#) la Referencia AWS SDK para Ruby de la API.

GetAccountPasswordPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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
  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
```

- Para obtener más información sobre la API, consulta [GetAccountPasswordPolicy](#) la Referencia AWS SDK para Ruby de la API.

GetPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [GetPolicy](#) la Referencia AWS SDK para Ruby de la API.

GetRole

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

SDK para Ruby

Note

Hay más información al respecto 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  
end
```

- Para obtener más información sobre la API, consulta [GetRole](#) la Referencia AWS SDK para Ruby de la API.

GetUser

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [GetUser](#) la Referencia AWS SDK para Ruby de la API.

ListAccessKeys

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

SDK para Ruby

 Note

Hay más información al respecto 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
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

```
@logger.info("Access key created for user '#{user_name}':  
#{access_key.access_key_id}")  
  access_key  
rescue Aws::IAM::Errors::LimitExceeded  
  @logger.error('Error creating access key: limit exceeded. Cannot create more.')  
  nil  
rescue StandardError => e  
  @logger.error("Error creating access key: #{e.message}")  
  nil  
end  
  
# Deactivates an access key  
#  
# @param user_name [String] The name of the user.  
# @param access_key_id [String] The ID for the access key.  
# @return [Boolean]  
def deactivate_access_key(user_name, access_key_id)  
  @iam_client.update_access_key(  
    user_name: user_name,  
    access_key_id: access_key_id,  
    status: 'Inactive'  
  )  
  true  
rescue StandardError => e  
  @logger.error("Error deactivating access key: #{e.message}")  
  false  
end  
  
# Deletes an access key  
#  
# @param user_name [String] The name of the user.  
# @param access_key_id [String] The ID for the access key.  
# @return [Boolean]  
def delete_access_key(user_name, access_key_id)  
  @iam_client.delete_access_key(  
    user_name: user_name,  
    access_key_id: access_key_id  
  )  
  true  
rescue StandardError => e  
  @logger.error("Error deleting access key: #{e.message}")  
  false  
end  
end
```

- Para obtener más información sobre la API, consulta [ListAccessKeys](#) la Referencia AWS SDK para Ruby de la API.

ListAccountAliases

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

SDK para Ruby

Note

Hay más información al respecto 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
end
```

```
end

# Creates an AWS account alias.
#
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
def create_account_alias(account_alias)
  @iam_client.create_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating account alias: #{e.message}")
  false
end

# Deletes an AWS account alias.
#
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
def delete_account_alias(account_alias)
  @iam_client.delete_account_alias(account_alias: account_alias)
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting account alias: #{e.message}")
  false
end
end
```

- Para obtener más información sobre la API, consulta [ListAccountAliases](#) la Referencia AWS SDK para Ruby de la API.

ListAttachedRolePolicies

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
end
```

```
@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  
end
```

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

- Para obtener más información sobre la API, consulta [ListAttachedRolePolicies](#) la Referencia AWS SDK para Ruby de la API.

ListGroups

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
  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
end

```

- Para obtener más información sobre la API, consulta [ListGroups](#) la Referencia AWS SDK para Ruby de la API.

ListPolicies

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

SDK para Ruby

Note

Hay más información al respecto 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
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 obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK para Ruby de la API.

ListRolePolicies

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListRolePolicies](#) la Referencia AWS SDK para Ruby de la API.

ListRoles

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListRoles](#) la Referencia AWS SDK para Ruby de la API.

ListSAMLProviders

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [lista SAMLProviders](#) en la referencia AWS SDK para Ruby de la API.

ListServerCertificates

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

SDK para Ruby

Note

Hay más información al respecto 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
  end
end

```

```
@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
  )
end
```

```

    @logger.info("Server certificate name updated from '#{current_name}' to
    '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

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

```

- Para obtener más información sobre la API, consulta [ListServerCertificates](#) la Referencia AWS SDK para Ruby de la API.

ListUsers

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListUsers](#) la Referencia AWS SDK para Ruby de la API.

PutUserPolicy

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [PutUserPolicy](#) la Referencia AWS SDK para Ruby de la API.

UpdateServerCertificate

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

SDK para Ruby

Note

Hay más información al respecto 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
    })
  end
end
```

```
    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}")
  end
end
```

```
    false
  end
end
```

- Para obtener más información sobre la API, consulta [UpdateServerCertificate](#) la Referencia AWS SDK para Ruby de la API.

UpdateUser

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [UpdateUser](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Kinesis con SDK para Ruby

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología 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.

SDK para Ruby

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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}"
    end
  end
end
```

```
# 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.

SDK para Ruby

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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 de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

CreateKey

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

SDK para Ruby

 Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateKey](#) la Referencia AWS SDK para Ruby de la API.

Decrypt

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

SDK para Ruby

Note

Hay más información al respecto 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 obtener información sobre la API, consulte [Decrypt](#) en la Referencia de la API de AWS SDK para Ruby .

Encrypt

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

SDK para 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 obtener información sobre la API, consulte [Encrypt](#) en la referencia de la API de AWS SDK para Ruby .

ReEncrypt

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

SDK para Ruby

 Note

Hay más información en 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 obtener más información sobre la API, consulta [ReEncrypt](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Lambda usando SDK para Ruby

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

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 distintas funciones de servicio, 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Lambda

El siguiente ejemplo de código muestra cómo empezar a utilizar Lambda.

SDK para Ruby

Note

Hay más información al respecto. 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 obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK para Ruby de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rol de IAM y una función de Lambda y, a continuación, cargar el código de 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](#).

SDK para Ruby

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Configurar los permisos de IAM de requisitos previos para que una función de Lambda pueda 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)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
  role_policy = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
```

```

        'Principal': { 'Service': 'lambda.amazonaws.com' },
        'Action': 'sts:AssumeRole'
      }
    ]
  }
}
role = @iam_client.create_role(
  role_name: iam_role_name,
  assume_role_policy_document: role_policy.to_json
)
@iam_client.attach_role_policy(
  {
    policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
    role_name: iam_role_name
  }
)
wait_for_role_to_exist(iam_role_name)
@logger.debug("Successfully created IAM role: #{role['role']['arn']}")
sleep(10)
[role, role_policy.to_json]
end

def destroy_iam_role(iam_role_name)
  @iam_client.detach_role_policy(
    {
      policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
      role_name: iam_role_name
    }
  )
  @iam_client.delete_role(role_name: iam_role_name)
  @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
end

def wait_for_role_to_exist(iam_role_name)
  @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
    w.max_attempts = 5
    w.delay = 5
  end
end
end

```

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.
#
# @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
end

```

```

Zip::File.open('lambda_function.zip', create: true) do |zipfile|
  zipfile.add('lambda_function.rb', "#{source_file}.rb")
end
@logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
File.read('lambda_function.zip').to_s
rescue StandardError => e
  @logger.error("There was an error creating deployment package:\n #{e.message}")
end

```

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.
# @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}")
end

```

```
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 listing functions:\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 obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Ruby .
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Acciones

CreateFunction

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

SDK para 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, :cloudwatch_client, :iam_client

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

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

```

        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 obtener más información sobre la API, consulta [CreateFunction](#) la Referencia AWS SDK para Ruby de la API.

DeleteFunction

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

SDK para Ruby

Note

Hay más información al respecto 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, :cloudwatch_client, :iam_client

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

```

```
@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 obtener más información sobre la API, consulta [DeleteFunction](#) la Referencia AWS SDK para Ruby de la API.

GetFunction

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

SDK para Ruby

Note

Hay más información al respecto 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, :cloudwatch_client, :iam_client

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

```

end

# 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 obtener más información sobre la API, consulta [GetFunction](#) la Referencia AWS SDK para Ruby de la API.

Invoke

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

SDK para Ruby

Note

Hay más información al respecto 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, :cloudwatch_client, :iam_client

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

```

```

    @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 obtener información sobre la API, consulte [Invocar](#) en la Referencia de la API de AWS SDK para Ruby .

ListFunctions

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

SDK para 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, :cloudwatch_client, :iam_client

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

```

```
# Lists the Lambda functions for the current account.
def list_functions
  functions = []
  @lambda_client.list_functions.each do |response|
    response['functions'].each do |function|
      functions.append(function['function_name'])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

- Para obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK para Ruby de la API.

UpdateFunctionCode

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

SDK para Ruby

Note

Hay más información al respecto 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, :cloudwatch_client, :iam_client

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

```

# 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
end

```

- Para obtener más información sobre la API, consulta [UpdateFunctionCode](#) la Referencia AWS SDK para Ruby de la API.

UpdateFunctionConfiguration

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

SDK para Ruby

Note

Hay más información al respecto 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, :cloudwatch_client, :iam_client

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

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
      function_name: function_name,
      environment: {
        variables: {
          'LOG_LEVEL' => log_level
        }
      }
    })

    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end

    rescue Aws::Lambda::Errors::ServiceException => e
      @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
    rescue Aws::Waiters::Errors::WaiterFailed => e
      @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
    end
  end
end

```

- Para obtener más información sobre la API, consulta [UpdateFunctionConfiguration](#) la Referencia AWS SDK para Ruby de la API.

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.

SDK para 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 ver el código fuente y las instrucciones de implementación, consulta el proyecto en [GitHub](#).

Servicios utilizados en este ejemplo

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

Ejemplos de tecnología sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Ruby

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función de 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,
    enableCleartextPlugin: 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.

SDK para Ruby

Note

Hay más información [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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

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

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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 desencadenado al recibir registros de una secuencia de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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
```

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

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

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de Amazon MSK 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}"
    end
  end
end
```

```
# 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 la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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}"
  end
end
```

```
    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
```

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

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SNS 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
end
```

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SQS 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.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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 parcial por lotes para las funciones de Lambda que reciben eventos de una secuencia 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.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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 los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. 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.

SDK para Ruby

Note

Hay más información [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de SQS 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
end
```

Ejemplos de Amazon MSK con SDK para Ruby

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes mediante el uso AWS SDK para Ruby de Amazon MSK.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología sin servidor

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

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

SDK para Ruby

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de Amazon MSK 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 con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para 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 distintas funciones de servicio, 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

DescribeVoices

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeVoices.

SDK para Ruby

Note

Hay más información al respecto. 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 => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Para obtener más información sobre la API, consulta [DescribeVoices](#) la Referencia AWS SDK para Ruby de la API.

ListLexicons

En el siguiente ejemplo de código, se muestra cómo utilizar ListLexicons.

SDK para Ruby

Note

Hay más información al respecto 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 => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Para obtener más información sobre la API, consulta [ListLexicons](#) la Referencia AWS SDK para Ruby de la API.

SynthesizeSpeech

En el siguiente ejemplo de código, se muestra cómo utilizar SynthesizeSpeech.

SDK para Ruby

 Note

Hay más información al respecto 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 => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- Para obtener más información sobre la API, consulta [SynthesizeSpeech](#) la Referencia AWS SDK para Ruby de la API.

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.

SDK para 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 ver el código fuente y las instrucciones de implementación, consulta el proyecto en [GitHub](#).

Servicios utilizados en este ejemplo

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

Ejemplos de Amazon RDS usando SDK para Ruby

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

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Amazon RDS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon RDS.

SDK para Ruby

 Note

Hay más información al respecto. 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 obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para Ruby API.

Acciones

CreateDBSnapshot

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBSnapshot.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Crear DBSnapshot](#) en la referencia de la AWS SDK para Ruby API.

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBInstances.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para Ruby API.

DescribeDBParameterGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameterGroups.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Describir DBParameter los grupos](#) en la referencia de la AWS SDK para Ruby API.

DescribeDBParameters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameters.

SDK para Ruby

 Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Describir DBParameters](#) en la referencia de la AWS SDK para Ruby API.

DescribeDBSnapshots

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBSnapshots.

SDK para Ruby

 Note

Hay más información al respecto 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 obtener más información sobre la API, consulta la [sección Describir DBSnapshots](#) en la referencia de la AWS SDK para Ruby API.

Ejemplos de tecnología sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Ruby

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función de 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 usando SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK para 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 distintas funciones de servicio, 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)

- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Amazon S3

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon S3.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK para Ruby de la API.

Conceptos básicos

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.

SDK para Ruby

Note

Hay más información al respecto 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: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

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

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
  destination

```

```

# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
def upload_file(bucket)
  File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
  s3_object = bucket.object(File.basename('demo.txt'))
  s3_object.upload_file('demo.txt')
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

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

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
  end
end

```

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

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Ruby .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Acciones

CopyObject

En el siguiente ejemplo de código, se muestra cómo utilizar CopyObject.

SDK para 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 = "amzn-s3-demo-bucket1"
```

```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

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)
  end
end

```

```

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

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [CopyObject](#) la Referencia AWS SDK para Ruby de la API.

CreateBucket

En el siguiente ejemplo de código, se muestra cómo utilizar CreateBucket.

SDK para Ruby

 Note

Hay más información al respecto 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
  end
end
```

```

    rescue Aws::Errors::ServiceError => e
      "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
    end
  end

  # Example usage:
  def run_demo
    region = "us-west-2"
    wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-#{Random.uuid}"))
    return unless wrapper.create?(region)

    puts "Created bucket #{wrapper.bucket.name}."
    puts "Your bucket's region is: #{wrapper.location}"
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [CreateBucket](#) la Referencia AWS SDK para Ruby de la API.

DeleteBucket

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucket.

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información sobre la API, consulta [DeleteBucket](#) la Referencia AWS SDK para Ruby de la API.

DeleteBucketCors

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucketCors.

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información sobre la API, consulta [DeleteBucketCors](#) la Referencia AWS SDK para Ruby de la API.

DeleteBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucketPolicy.

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información sobre la API, consulta [DeleteBucketPolicy](#) la Referencia AWS SDK para Ruby de la API.

DeleteObjects

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteObjects.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [DeleteObjects](#) la Referencia AWS SDK para Ruby de la API.

GetBucketCors

En el siguiente ejemplo de código, se muestra cómo utilizar GetBucketCors.

SDK para Ruby

Note

Hay más información al respecto 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 cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
      nil
    end
  end
end
```

- Para obtener más información sobre la API, consulta [GetBucketCors](#) la Referencia AWS SDK para Ruby de la API.

GetBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar GetBucketPolicy.

SDK para Ruby

Note

Hay más información al respecto 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 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 obtener más información sobre la API, consulta [GetBucketPolicy](#) la Referencia AWS SDK para Ruby de la API.

GetObject

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObject`.

SDK para Ruby

Note

Hay más información al respecto 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 = "amzn-s3-demo-bucket"
object_key = "my-object.txt"
target_path = "my-object-as-file.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

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 object
    @object.get
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end

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

```

```

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [GetObject](#) la Referencia AWS SDK para Ruby de la API.

HeadObject

En el siguiente ejemplo de código, se muestra cómo utilizar HeadObject.

SDK para Ruby

Note

Hay más información al respecto 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 = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [HeadObject](#) la Referencia AWS SDK para Ruby de la API.

ListBuckets

En el siguiente ejemplo de código, se muestra cómo utilizar ListBuckets.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK para Ruby de la API.

ListObjectsV2

En el siguiente ejemplo de código, se muestra cómo utilizar ListObjectsV2.

SDK para Ruby

 Note

Hay más información al respecto 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 = "amzn-s3-demo-bucket"
```

```

    wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
    count = wrapper.list_objects(25)
    puts "Listed #{count} objects."
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta la [ListObjectsversión 2](#) en la referencia de la AWS SDK para Ruby API.

PutBucketCors

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketCors.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [PutBucketCors](#) la Referencia AWS SDK para Ruby de la API.

PutBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketPolicy.

SDK para Ruby

Note

Hay más información al respecto 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 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 obtener más información sobre la API, consulta [PutBucketPolicy](#) la Referencia AWS SDK para Ruby de la API.

PutBucketWebsite

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketWebsite.

SDK para Ruby

Note

Hay más información al respecto 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 = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [PutBucketWebsite](#) la Referencia AWS SDK para Ruby de la API.

PutObject

En el siguiente ejemplo de código, se muestra cómo utilizar PutObject.

SDK para Ruby

Note

Hay más información al respecto 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 = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

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 = "amzn-s3-demo-bucket"
object_key = "my-object-key"
file_path = "my-local-file.txt"

wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
success = wrapper.put_object(file_path)
return unless success

puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

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 = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

```

```

    wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
object_content))
    return unless wrapper.put_object_encrypted(object_content, encryption)

    puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
#{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Para obtener más información sobre la API, consulta [PutObject](#) la Referencia AWS SDK para Ruby de la API.

Escenarios

Crear una URL prefirmada

En el siguiente ejemplo de código se muestra cómo crear una URL prefirmada para Amazon S3 y cargar un objeto.

SDK para Ruby

Note

Hay más información al respecto 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)
end

```

```
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
    #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
      'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Ejemplos de tecnología 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 la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología 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
```

Ejemplos de Amazon SES con SDK para Ruby

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

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

GetIdentityVerificationAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar `GetIdentityVerificationAttributes`.

SDK para Ruby

Note

Hay más información al respecto 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
    puts email if status == 'Success'
  end
end
```

- Para obtener más información sobre la API, consulta [GetIdentityVerificationAttributes](#) la Referencia AWS SDK para Ruby de la API.

ListIdentities

En el siguiente ejemplo de código, se muestra cómo utilizar `ListIdentities`.

SDK para Ruby

Note

Hay más información al respecto 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]
  })
end
```

```
status = attrs.verification_attributes[email].verification_status

# Display email addresses that have been verified
puts email if status == 'Success'
end
```

- Para obtener más información sobre la API, consulta [ListIdentities](#) la Referencia AWS SDK para Ruby de la API.

SendEmail

En el siguiente ejemplo de código, se muestra cómo utilizar SendEmail.

SDK para Ruby

Note

Hay más información al respecto 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 => e
  puts "Email not sent. Error message: #{e}"
end
```

- Para obtener más información sobre la API, consulta [SendEmail](#) la Referencia AWS SDK para Ruby de la API.

VerifyEmailIdentity

En el siguiente ejemplo de código, se muestra cómo utilizar VerifyEmailIdentity.

SDK para Ruby

Note

Hay más información al respecto 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
  })
end
```

```
puts "Email sent to #{recipient}"

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

- Para obtener más información sobre la API, consulta [VerifyEmailIdentity](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Amazon SES API v2 con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Ruby API v2 de Amazon SES.

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

SendEmail

En el siguiente ejemplo de código, se muestra cómo utilizar `SendEmail`.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [SendEmail](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Amazon SNS usando SDK para Ruby

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

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)
- [Ejemplos de tecnología sin servidor](#)

Acciones

CreateTopic

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTopic.

SDK para Ruby

Note

Hay más información al respecto. 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
end

```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para Ruby](#).
- Para obtener más información sobre la API, consulta [CreateTopic](#) la Referencia AWS SDK para Ruby de la API.

ListSubscriptions

En el siguiente ejemplo de código, se muestra cómo utilizar ListSubscriptions.

SDK para Ruby

Note

Hay más información al respecto 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 más información, consulte la [Guía para desarrolladores de AWS SDK para Ruby](#).
- Para obtener más información sobre la API, consulta [ListSubscriptions](#) la Referencia AWS SDK para Ruby de la API.

ListTopics

En el siguiente ejemplo de código, se muestra cómo utilizar ListTopics.

SDK para Ruby

Note

Hay más información al respecto 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.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para Ruby](#).

- Para obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK para Ruby de la API.

Publish

En el siguiente ejemplo de código, se muestra cómo utilizar Publish.

SDK para Ruby

Note

Hay más información al respecto 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
end

```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para Ruby](#).
- Para obtener detalles sobre la API, consulte [Publish](#) en la Referencia de la API de AWS SDK para Ruby .

SetTopicAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar SetTopicAttributes.

SDK para 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 más información, consulte la [Guía para desarrolladores de AWS SDK para Ruby](#).
- Para obtener más información sobre la API, consulta [SetTopicAttributes](#) la Referencia AWS SDK para Ruby de la API.

Subscribe

En el siguiente ejemplo de código, se muestra cómo utilizar Subscribe.

SDK para Ruby

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Suscriba 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 para Ruby](#).
- Para obtener detalles sobre la API, consulte [Subscribe](#) en la Referencia de la API de AWS SDK para Ruby .

Ejemplos de tecnología sin servidor

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

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SNS 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
```

Ejemplos de Amazon SQS usando SDK para Ruby

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

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

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)
- [Ejemplos de tecnología sin servidor](#)

Acciones

ChangeMessageVisibility

En el siguiente ejemplo de código, se muestra cómo utilizar ChangeMessageVisibility.

SDK para Ruby

Note

Hay más información al respecto. 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 up to 10 messages
  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })
```

```
puts "Before attempting to change message visibility timeout: received
#{receive_message_result_before.messages.count} message(s)."
```

```
receive_message_result_before.messages.each do |message|
  sqs.change_message_visibility({
    queue_url: queue_url,
    receipt_handle: message.receipt_handle,
    visibility_timeout: 30 # This message will not
be visible for 30 seconds after first receipt.
  })
end
```

```
# Try to retrieve the original messages after setting their visibility timeout.
receive_message_result_after = sqs.receive_message({
  queue_url: queue_url,
  max_number_of_messages: 10
})
```

```
puts "\nAfter attempting to change message visibility timeout: received
#{receive_message_result_after.messages.count} message(s)."
```

```
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
exist."
end
```

- Para obtener más información sobre la API, consulta [ChangeMessageVisibility](#) la Referencia AWS SDK para Ruby de la API.

CreateQueue

En el siguiente ejemplo de código, se muestra cómo utilizar CreateQueue.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [CreateQueue](#) la Referencia AWS SDK para Ruby de la API.

DeleteQueue

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteQueue.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [DeleteQueue](#) la Referencia AWS SDK para Ruby de la API.

ListQueues

En el siguiente ejemplo de código, se muestra cómo utilizar ListQueues.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK para Ruby de la API.

ReceiveMessage

En el siguiente ejemplo de código, se muestra cómo utilizar ReceiveMessage.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [ReceiveMessage](#) la Referencia AWS SDK para Ruby de la API.

SendMessage

En el siguiente ejemplo de código, se muestra cómo utilizar SendMessage.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [SendMessage](#) la Referencia AWS SDK para Ruby de la API.

SendMessageBatch

En el siguiente ejemplo de código, se muestra cómo utilizar SendMessageBatch.

SDK para Ruby

Note

Hay más información al respecto 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
end
```

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

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
  #{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- Para obtener más información sobre la API, consulta [SendMessageBatch](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de tecnología sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Ruby

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SQS 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 Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. 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.

SDK para Ruby

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de SQS 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
end
```

AWS STS ejemplos de uso de SDK for Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para 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 distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

AssumeRole

En el siguiente ejemplo de código, se muestra cómo utilizar AssumeRole.

SDK para Ruby

Note

Hay más información al respecto 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 obtener más información sobre la API, consulta [AssumeRole](#) la Referencia AWS SDK para Ruby de la API.

Ejemplos de Amazon Textract con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el 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.

SDK para 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 con SDK para Ruby

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para 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.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el 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.

SDK para 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

Migración de AWS SDK para Ruby versión 1 o 2 a AWS SDK para Ruby versión 3

Este tema incluye detalles para ayudarlo 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:
 - `:use_ssl` Se ha eliminado. 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`
 - Se ha agregado `:ssl_ca_store`.
 - `:endpoint` Ahora 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.

Seguridad para AWS SDK for 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

- [Protección de datos en AWS SDK for Ruby](#)
- [Identity and Access Management para AWS SDK for Ruby](#)
- [Validación de conformidad para AWS SDK for Ruby](#)
- [Resiliencia para AWS SDK for Ruby](#)
- [Seguridad de infraestructura para AWS SDK for Ruby](#)
- [Imponer una versión mínima de TLS en el AWS SDK para Ruby](#)
- [Migración del cliente de cifrado Amazon S3 \(V1 a V2\)](#)
- [Migración del cliente de cifrado Amazon S3 \(V2 a V3\)](#)

Protección de datos en AWS SDK for 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. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres 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 las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el [Modelo de responsabilidad compartida de AWS y GDPR](#) en el Blog de seguridad de AWS .

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:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte [Cómo trabajar con CloudTrail senderos](#) en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 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, la API o. AWS CLI AWS SDKs Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Identity and Access Management para AWS SDK for 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. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Servicios de AWS. El IAM es un Servicio de AWS servicio que puede utilizar sin coste adicional.

Para poder acceder a AWS SDK for Ruby AWS, necesitas una AWS cuenta y AWS credenciales. Para aumentar la seguridad de su cuenta de AWS , le recomendamos que utilice un usuario de IAM a la hora de proporcionar credenciales de acceso en lugar de usar las credenciales de su cuenta de AWS .

Para obtener más información acerca de cómo trabajar con IAM, consulte [IAM](#).

Para obtener información general sobre los usuarios de IAM y por qué son importantes para la seguridad de su cuenta, consulte [Credenciales de seguridad de AWS](#) en la [Referencia general de Amazon Web Services](#).

AWS El SDK for 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 para AWS SDK for Ruby

AWS El SDK for 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](#).

Audidores 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, FedRAMP, HIPAA y otros. AWS proporciona una lista actualizada con frecuencia de los programas de cumplimiento específicos dentro del ámbito de aplicación Servicios de AWS 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 de cumplimiento al usar AWS SDK for Ruby para acceder a un 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 de un Servicio de AWS está sujeto al cumplimiento de normas como HIPAA, PCI o FedRAMP, proporciona recursos que le ayudarán a: AWS

- 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
- [Referencia de servicios válidos de HIPAA](#): muestra una lista con los servicios válidos de HIPAA. No todos cumplen con los requisitos de la HIPAA. Servicios de AWS
- [AWS Recursos de cumplimiento](#): una colección de libros de trabajo y guías que pueden aplicarse a su industria 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 para AWS SDK for 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 El SDK for 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 para AWS SDK for Ruby

AWS El SDK for 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 versión mínima de TLS en el AWS SDK para Ruby

La comunicación entre el AWS SDK para Ruby y AWS está protegida mediante Secure Sockets Layer (SSL) o Transport Layer Security (TLS). Todas las versiones de SSL y las versiones de TLS anteriores a la 1.2 presentan vulnerabilidades que pueden comprometer la seguridad de la comunicación. AWS Por este motivo, debes asegurarte de usar el AWS SDK para Ruby con una versión de Ruby compatible con la versión 1.2 o posterior de TLS.

Ruby usa la biblioteca OpenSSL para proteger las conexiones HTTP. Las versiones compatibles de Ruby (1.9.3 y posteriores) instaladas mediante los [administradores de paquetes](#) del sistema (yum, apt, etc.), un [instalador oficial](#) o [administradores](#) de Ruby (rbenv, RVM, etc.) habitualmente incorporan OpenSSL 1.0.1 o posterior, que es compatible con TLS 1.2.

Cuando se usa con una versión compatible de Ruby con OpenSSL 1.0.1 o posterior, el SDK para AWS Ruby prefiere TLS 1.2 y usa la última versión de SSL o TLS compatible tanto con el cliente como con el servidor. Siempre es al menos para TLS 1.2. Servicios de AWS(El SDK utiliza la clase `Net::HTTP` de Ruby con `use_ssl=true`)

Comprobar la versión de OpenSSL

Para asegurarse de que su instalación de Ruby está usando OpenSSL 1.0.1 o posterior, introduzca el siguiente comando.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Otra forma de obtener la versión de OpenSSL es buscar directamente el ejecutable de `openssl`. Primero, localice el ejecutable adecuado mediante el siguiente comando.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

La salida debe tener `--with-openssl-dir=/path/to/openssl` que indique la ubicación de la instalación de OpenSSL. 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.

Actualizar la compatibilidad con TLS

Si la versión de OpenSSL utilizada por Ruby es inferior a 1.0.1, actualice la instalación de Ruby u OpenSSL mediante el administrador de paquetes de su sistema, el instalador de Ruby o el administrador de Ruby tal como se describe en la [guía de instalación](#) de Ruby. Si va a instalar Ruby [desde el código fuente](#), instale primero la [última versión de OpenSSL](#) y pase `--with-openssl-dir=/path/to/upgraded/openssl` cuando ejecute `./configure`.

Migración del cliente de cifrado Amazon S3 (V1 a V2)

Note

Si utiliza la versión 2 del cliente de cifrado S3 y desea migrar a la versión 3, consulte [Migración del cliente de cifrado Amazon S3 \(V2 a V3\)](#).

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. Primero, implementa una versión actualizada del AWS SDK for Ruby en tu 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 tu aplicación usa varios AWS SDKs, debes actualizar cada SDK por separado.
2. Migre 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 V1 a la versión más reciente del SDK. 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. Consulta los detalles de cada versión principal del AWS SDK for Ruby a continuación.

Actualización AWS del SDK para Ruby versión 3

La versión 3 es la última versión del AWS SDK para 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 AWS del SDK para Ruby versión 2

La versión 2 del AWS SDK for 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, lea todo el objeto hasta el final antes de empezar a utilizar los datos descifrados. Esto se hace 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 lo está utilizando AWS KMS, debe estar configurado en `:kms_context`. Si utiliza una clave simétrica (AES), esta se debe establecer en `:aes_gcm`. Si utiliza una clave asimétrica (RSA), esta se debe establecer 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 `kms_key_id` versión 1 `Encryption::Client`, no se proporcionó al 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, en `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_cmek` en la creación del cliente o establezca `kms_allow_decrypt_with_any_cmek: true` en llamadas de `get_object`.

Ejemplo: Uso de una clave simétrica (AES)

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
```

AWS KMS Ejemplo: usar 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_cmek:
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_cmek, # set kms_key_id to allow all get_object
requests to use any cmk
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
the V1 encryption client
)
resp = client.get_object(bucket: bucket, key: key) # No change
```

Migración del cliente de cifrado Amazon S3 (V2 a V3)

Note

Si utiliza la V1 del cliente de cifrado S3, primero debe migrar a la V2 antes de migrar a la V3. Consulte [Migración del cliente de cifrado Amazon S3 \(V1 a V2\)](#) para obtener instrucciones sobre cómo migrar de la V1 a la V2.

En este tema se muestra cómo migrar las aplicaciones de la versión 2 (V2) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) a la versión 3 (V3) y cómo garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración. La versión 3 presenta el AES GCM con políticas clave de compromiso y compromiso para mejorar la seguridad y la protección contra la manipulación de las claves de datos.

Información general sobre la migración

La versión 3 del cliente de cifrado Amazon S3 presenta AES GCM con Key Commitment para mejorar la seguridad. Este nuevo algoritmo de cifrado proporciona protección contra la manipulación de las claves de datos y garantiza la integridad de los datos cifrados. La migración a la versión 3 requiere una planificación cuidadosa para mantener la disponibilidad de las aplicaciones y la accesibilidad de los datos durante todo el proceso.

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. Primero, implementa una versión actualizada del AWS SDK for Ruby en tu aplicación. Esto permitirá a los clientes de cifrado V2 existentes descifrar los objetos escritos por los nuevos clientes V3. Si su aplicación usa varios AWS SDKs, debe actualizar cada SDK por separado.
2. Migre los clientes de cifrado y descifrado a la versión 3. Una vez que todos sus clientes de cifrado V2 puedan leer nuevos formatos, puede migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones V3. Esto incluye configurar las políticas de compromiso y actualizar el código para utilizar las nuevas opciones de configuración del cliente.

Si aún no ha migrado de la V1 a la V2, primero debe completar la migración. Consulte [Migración del cliente de cifrado Amazon S3 \(V1 a V2\)](#) para obtener instrucciones detalladas sobre la migración de la V1 a la V2.

Comprensión de las funciones de la V3

La versión 3 del cliente de cifrado Amazon S3 presenta dos características de seguridad clave: políticas de compromiso y AES GCM con compromiso clave. Comprender estas características es esencial para planificar su estrategia de migración y garantizar la seguridad de sus datos cifrados.

Políticas de compromiso

Las políticas de compromiso controlan la forma en que el cliente de cifrado gestiona el compromiso de claves durante las operaciones de cifrado y descifrado. El compromiso clave garantiza que los datos cifrados solo se puedan descifrar con la clave exacta que se utilizó para cifrarlos, lo que los protege contra ciertos tipos de ataques criptográficos.

El cliente de cifrado V3 admite tres opciones de política de compromiso:

FORBID_ENCRYPT_ALLOW_DECRYPT

Esta política cifra los objetos sin compromiso de clave y permite descifrar ambos objetos con o sin compromiso de clave.

- Comportamiento de cifrado: los objetos se cifran sin compromiso de clave y utilizan el mismo conjunto de algoritmos que V2.
- Comportamiento de descifrado: puede descifrar objetos cifrados con o sin compromiso de clave.
- Implicaciones de seguridad: esta política no impone un compromiso clave y puede permitir la manipulación. Los objetos cifrados con esta política no se benefician de las protecciones de seguridad mejoradas que conlleva un compromiso clave. Utilice esta política solo durante la migración cuando necesite mantener la compatibilidad con el comportamiento de cifrado de la versión 2.
- Compatibilidad de versiones: los objetos cifrados con esta política pueden leerse en todas las implementaciones V2 y V3 del cliente de cifrado S3.

REQUIRE_ENCRYPT_ALLOW_DECRYPT

Esta política cifra los objetos con compromiso de clave y permite descifrar ambos objetos con o sin compromiso de clave.

- Comportamiento de cifrado: los objetos se cifran con un compromiso de clave mediante el AES GCM con un compromiso de clave.

- Comportamiento de descifrado: puede descifrar objetos cifrados con o sin compromiso de clave, lo que proporciona compatibilidad con versiones anteriores.
- Implicaciones de seguridad: los objetos nuevos se benefician de la protección con compromiso clave, mientras que los objetos existentes sin compromiso clave aún se pueden leer. Esto proporciona un equilibrio entre la seguridad y la compatibilidad con versiones anteriores durante la migración.
- Compatibilidad de versiones: los objetos cifrados con esta política solo los pueden leer las implementaciones V3 y V2 más recientes del cliente de cifrado S3.

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

Esta política cifra los objetos con un compromiso de clave y solo permite descifrar los objetos que se cifraron con un compromiso de clave.

- Comportamiento de cifrado: los objetos se cifran con un compromiso de clave mediante el AES GCM con un compromiso de clave.
- Comportamiento de descifrado: solo se pueden descifrar los objetos que se cifraron con un compromiso de clave. Los intentos de descifrar objetos sin un compromiso de clave fallarán.
- Implicaciones de seguridad: esta política proporciona el más alto nivel de seguridad al imponer un compromiso clave en todas las operaciones. Utilice esta política solo después de que todos los objetos se hayan vuelto a cifrar con la clave y de que todos los clientes se hayan actualizado a la versión 3.
- Compatibilidad de versiones: los objetos cifrados con esta política solo pueden leerse en la versión 3 y las implementaciones más recientes del cliente de cifrado S3 en la versión 2. Esta política también impide leer los objetos cifrados por los clientes V2 o V1.

Note

Cuando planifique la migración, comience por mantener la compatibilidad con versiones anteriores y, REQUIRE_ENCRYPT_ALLOW_DECRYPT al mismo tiempo, obtenga las ventajas de seguridad que supone un compromiso clave con los nuevos objetos. Solo muévase a una REQUIRE_ENCRYPT_REQUIRE_DECRYPT vez que todos los objetos se hayan vuelto a cifrar y todos los clientes se hayan actualizado a la versión 3.

AES GCM con un compromiso clave

El AES GCM con Key Commitment (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) es un nuevo algoritmo de cifrado introducido en la versión 3 que proporciona una mayor seguridad al proteger contra la manipulación de las claves de datos. Entender cómo funciona este algoritmo y cuándo se aplica es importante para planificar la migración.

¿En qué `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` se diferencia de los algoritmos anteriores

Las versiones anteriores del cliente de cifrado S3 utilizaban AES CBC o AES GCM sin el compromiso de cifrar la clave de datos en los archivos de instrucciones.

`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` añade un compromiso criptográfico al proceso de cifrado, que vincula los datos cifrados a una clave específica. Esto evita que un atacante altere la clave de datos cifrada del archivo de instrucciones y provoque que el cliente descifre los datos con una clave incorrecta.

Sin un compromiso clave, es posible que un atacante modifique la clave de datos cifrados de un archivo de instrucciones para convertirla en una clave diferente, lo que podría provocar el acceso no autorizado o la corrupción de los datos. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` evita este ataque al garantizar que la clave de datos cifrados solo pueda descifrarse con la clave original que se utilizó durante el cifrado.

Compatibilidad de versiones

Los objetos cifrados solo se `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` pueden descifrar mediante las implementaciones de la versión 3 del cliente de cifrado S3 y determinadas versiones de transición de la versión 2, que incluyen soporte para leer los formatos de la versión 3. Los clientes de la versión 2 que no admiten esta transición no pueden descifrar los archivos de instrucciones cifrados con `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`.

Warning

Antes de habilitar el cifrado `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (mediante políticas de uso `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o de `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` compromiso), asegúrese de que todos los clientes que necesiten leer los objetos cifrados se hayan actualizado a la versión 3 o a una versión de transición que admita los formatos de la versión 3. Si algún

cliente de la versión 2 sin soporte de transición intenta leer los objetos cifrados con ellosALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY, el descifrado fallará.

Durante la migración, puede utilizar la política de FORBID_ENCRYPT_ALLOW_DECRYPT compromiso para seguir cifrando sin embargo y, al ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY mismo tiempo, permitir que sus clientes de la versión 3 lean los objetos cifrados con el compromiso de clave. Esto proporciona una ruta de migración segura en la que primero se actualizan todos los lectores y, a continuación, se pasa al cifrado con el compromiso de clave.

Actualizar los clientes existentes para leer nuevos formatos

El cliente de cifrado V3 utiliza algoritmos de cifrado y funciones de compromiso clave que los clientes V2 no admiten de forma predeterminada. El primer paso de la migración consiste en actualizar los clientes de descifrado de la versión 2 a una versión del AWS SDK para Ruby que pueda leer los objetos cifrados de la versión 3. Tras completar este paso, los clientes V2 de tu aplicación podrán descifrar los objetos cifrados por los clientes de cifrado V3.

Para leer los objetos cifrados por los clientes de la versión 3 (aquellos que utilizan políticas REQUIRE_ENCRYPT_ALLOW_DECRYPT o se REQUIRE_ENCRYPT_REQUIRE_DECRYPT comprometen a utilizarlas), debe utilizar la versión 1.93.0 o posterior de la gema. `aws-sdk-s3` Esta versión incluye soporte para descifrar objetos cifrados con AES GCM con Key Commitment.

Instalación desde la línea de comandos

Para los proyectos que instalan la `aws-sdk-s3` gema desde la línea de comandos, utilice la opción de versión para comprobar que está instalada la versión mínima de la 1.208.0.

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

Uso de archivos Gemfile

Para los proyectos que usan un Gemfile para administrar las dependencias, establece la versión mínima de la gema en 1.208.0. `aws-sdk-s3` Por ejemplo:

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. Modifica tu Gemfile para especificar la versión mínima.

2. Ejecuta `bundle update aws-sdk-s3` para actualizar la gema.
3. Para comprobar su versión, ejecute `bundle info aws-sdk-s3`.

Note

Tras actualizar a la última versión, sus clientes de cifrado V2 existentes podrán descifrar los objetos cifrados por los clientes V3. Sin embargo, seguirán cifrando los objetos nuevos mediante los algoritmos de la V2 hasta que los migre a la V3, tal y como se describe en la siguiente sección.

Migre los clientes de cifrado y descifrado a la V3

Tras actualizar sus clientes para que lean los nuevos formatos de cifrado, puede actualizar sus aplicaciones a los clientes de cifrado y descifrado de la V3. Los siguientes pasos le muestran cómo migrar correctamente su código de la V2 a la V3.

Antes de actualizar el código para usar el cliente de cifrado V3, asegúrese de haber seguido los pasos anteriores y de utilizar la `aws-sdk-s3` gema en la versión 1.93.0 o posterior.

Note

Al descifrar con AES-GCM, lea todo el objeto hasta el final antes de empezar a utilizar los datos descifrados. Esto se hace para verificar que el objeto no se ha modificado desde que se cifró.

Configuración de los clientes V3

El cliente de cifrado V3 presenta nuevas opciones de configuración que controlan el comportamiento del compromiso de claves y la compatibilidad con versiones anteriores. Comprender estas opciones es esencial para una migración exitosa.

`política_compromiso`

El `commitment_policy` parámetro controla la forma en que el cliente de cifrado gestiona la asignación de claves durante las operaciones de cifrado y descifrado. Esta es la opción de configuración más importante para los clientes de la versión 3.

- `:require_encrypt_allow_decrypt`- Cifra los objetos nuevos con un compromiso de clave y permite descifrar objetos con o sin compromiso de clave. Esta es la configuración recomendada para la migración, ya que proporciona una seguridad mejorada para los objetos nuevos y, al mismo tiempo, mantiene la compatibilidad con versiones anteriores de los objetos V2 existentes.
- `:forbid_encrypt_allow_decrypt`- Cifra los objetos nuevos sin compromiso de clave (mediante algoritmos V2) y permite descifrar objetos con o sin compromiso de clave. Utilice esta configuración solo si necesita mantener el comportamiento de cifrado de la versión 2 durante la migración, por ejemplo, cuando algunos clientes aún no pueden leer los objetos cifrados de la versión 3.
- `:require_encrypt_require_decrypt`- Cifra los objetos nuevos con un compromiso de clave y solo permite descifrar los objetos que se cifraron con un compromiso de clave. Utilice esta configuración solo después de que todos los objetos se hayan vuelto a cifrar con la clave de confirmación y de que todos los clientes se hayan actualizado a la versión 3.

perfil de seguridad

El `security_profile` parámetro determina la compatibilidad con la lectura de objetos escritos por versiones anteriores del cliente de cifrado. Este parámetro es esencial para mantener la compatibilidad con versiones anteriores durante la migración.

- `:v3_and_legacy`- Permite al cliente V3 descifrar objetos cifrados por los clientes de cifrado V1 y V2. Utilice esta configuración durante la migración para asegurarse de que sus clientes V3 puedan leer todos los objetos cifrados existentes.
- `:v3`- Permite al cliente V3 descifrar los objetos cifrados únicamente por los clientes de cifrado V2. Utilice esta configuración si ya ha migrado todos los objetos V1 al formato V2.
- Si no se especifica, el cliente solo descifrá los objetos cifrados por los clientes V3. Úselo solo para el desarrollo de nuevas aplicaciones donde no existan objetos heredados.

envelope_location

El `envelope_location` parámetro determina dónde se almacenan los metadatos de cifrado (incluida la clave de datos cifrados). Este parámetro afecta a los objetos que AES GCM protege con un compromiso clave.

- `:metadata(Predeterminado)`: almacena los metadatos de cifrado en los encabezados de metadatos del objeto S3. Este es el comportamiento predeterminado y se recomienda para la

mayoría de los casos de uso. Cuando se utiliza el almacenamiento de metadatos, no se aplica el AES GCM con compromiso clave.

- `:instruction_file`- Almacena los metadatos de cifrado en un objeto S3 independiente (archivo de instrucciones) con un sufijo configurable. Cuando se utilizan archivos de instrucciones, el AES GCM con Key Commitment protege la clave de datos cifrados contra la manipulación. Utilice esta configuración si necesita la seguridad adicional que proporciona el compromiso de clave para la propia clave de datos.

Si lo utiliza `:instruction_file`, puede especificar opcionalmente el `instruction_file_suffix` parámetro para personalizar el sufijo utilizado en los objetos del archivo de instrucciones. El sufijo predeterminado es `.instruction`

Cuándo utilizar cada opción de configuración

Durante la migración, siga esta estrategia de configuración recomendada:

1. Migración inicial: defina `commitment_policy: :require_encrypt_allow_decrypt` y `security_profile: :v3_and_legacy`. Esto permite a sus clientes de V3 cifrar nuevos objetos con un compromiso de clave y, al mismo tiempo, poder descifrar todos los objetos V1 y V2 existentes.
2. Una vez que se hayan actualizado todos los clientes: continúe utilizando `commitment_policy: :require_encrypt_allow_decrypt` y `security_profile: :v3_and_legacy` hasta que haya vuelto a cifrar todos los objetos que necesiten protección mediante compromiso de clave.
3. Aplicación completa de la versión 3: solo después de que todos los objetos se hayan vuelto a cifrar con la clave de confirmación y ya no sea necesario leer los objetos de las versiones 1/V2, puede cambiar al `security_profile` parámetro `commitment_policy: :require_encrypt_require_decrypt` y eliminarlo (o configurarlo `:v2` si aún existen objetos con la versión V2).

`envelope_location` Pues sigue usando tu método de almacenamiento actual (`:metadata` o `:instruction_file`) a menos que tengas un motivo específico para cambiarlo. Si actualmente utiliza el almacenamiento de metadatos y desea la seguridad adicional de AES GCM con un compromiso clave para la clave de datos, puede cambiarse a uno `:instruction_file`, pero tenga en cuenta que para ello será necesario actualizar todos los clientes que lean estos objetos.

Migre los clientes de cifrado y descifrado a la versión 3

Tras actualizar sus clientes para que lean los nuevos formatos de cifrado, puede actualizar sus aplicaciones a los clientes de cifrado y descifrado de la V3. Los siguientes ejemplos muestran cómo migrar correctamente el código de la V2 a la V3.

Uso de clientes de cifrado V3

Antes de la migración (V2)

```
require 'aws-sdk-s3'

# Create V2 encryption client with KMS
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy,
  commitment_policy: :forbid_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Durante la migración (versión 3 con compatibilidad con versiones anteriores)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt
)

# Encrypt and upload object
```

```
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Después de la migración (V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3,
  # Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
  # This encrypts with key commitment and does not decrypt V2 objects
  commitment_policy: :require_encrypt_require_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

La diferencia clave en la V3 es la adición del parámetro. `commitment_policy` Al configurarlo, se `:require_encrypt_require_decrypt` garantiza que los objetos nuevos se cifren con un compromiso de clave y que el cliente solo descifre los objetos cifrados con ese compromiso, lo que proporciona una mayor seguridad contra la manipulación de las claves de datos.

La `put_object` llamada en sí misma permanece inalterada. Todas las mejoras de seguridad se configuran a nivel de cliente.

Ejemplos adicionales

En esta sección se proporcionan ejemplos adicionales de escenarios de migración específicos y opciones de configuración que pueden resultar útiles durante la migración de la versión 2 a la versión 3.

Archivo de instrucciones versus almacenamiento de metadatos

El cliente de cifrado de S3 puede almacenar los metadatos de cifrado (incluida la clave de datos cifrados) en dos ubicaciones diferentes: en los encabezados de metadatos del objeto S3 o en un archivo de instrucciones independiente. La elección del método de almacenamiento afecta a los objetos que se benefician del AES GCM con la protección Key Commitment.

Almacenamiento de metadatos (predeterminado)

De forma predeterminada, el cliente de cifrado almacena los metadatos de cifrado en los encabezados de metadatos del objeto S3. Este es el enfoque recomendado para la mayoría de los casos de uso, ya que conserva los metadatos de cifrado con el objeto y no requiere la administración de objetos separados del archivo de instrucciones.

```
require 'aws-sdk-s3'

# Create V3 encryption client with metadata storage (default)
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :metadata # Explicitly set to metadata (this is the default)
)

# Encrypt and upload object
# Encryption metadata is stored in the object's metadata headers
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')
```

Cuando se utiliza el almacenamiento de metadatos, el AES GCM con clave de compromiso no se aplica a la clave de datos cifrados. Sin embargo, el cifrado del contenido sigue beneficiándose del compromiso de clave cuando se utiliza `commitment_policy: :require_encrypt_allow_decrypt` o `:require_encrypt_require_decrypt`.

Almacenamiento de archivos de instrucciones

Como alternativa, puede configurar el cliente de cifrado para almacenar los metadatos de cifrado en un objeto S3 independiente denominado archivo de instrucciones. Al utilizar los archivos de instrucciones con la versión V3, la clave de datos cifrada está protegida por AES GCM con el

compromiso de clave, lo que proporciona una seguridad adicional contra la manipulación de las claves de datos.

```
require 'aws-sdk-s3'

# Create V3 encryption client with instruction file storage
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :instruction_file, # Store metadata in separate instruction file
  instruction_file_suffix: '.instruction' # Optional: customize the suffix (default is
  '.instruction')
)

# Encrypt and upload object
# Encryption metadata is stored in a separate object: 'my-object.instruction'
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# When retrieving the object, the client automatically reads the instruction file
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

Cuando se utiliza `envelope_location: :instruction_file`, el cliente de cifrado crea dos objetos S3:

1. El objeto de datos cifrado (por ejemplo, `my-object`)
2. El archivo de instrucciones que contiene los metadatos de cifrado (por ejemplo, `my-object.instruction`)

El `instruction_file_suffix` parámetro permite personalizar el sufijo utilizado en los archivos de instrucciones. El valor predeterminado es `.instruction`.

Cuándo usar cada método de almacenamiento

- Utilice el almacenamiento de metadatos en la mayoría de los escenarios. Simplifica la administración de objetos, ya que los metadatos de cifrado viajan con el objeto.

- Utilice el almacenamiento de archivos de instrucciones cuando el tamaño de los metadatos del objeto sea un problema o cuando necesite separar los metadatos de cifrado del objeto cifrado. Tenga en cuenta que el uso de los archivos de instrucciones requiere administrar dos objetos S3 (el objeto cifrado y su archivo de instrucciones) en lugar de uno.

 Warning

Si cambia del almacenamiento de metadatos al almacenamiento de archivos de instrucciones (o viceversa), los clientes configurados con el nuevo método de almacenamiento no podrán leer los objetos existentes cifrados con el método de almacenamiento anterior. Planifique cuidadosamente el método de almacenamiento y mantenga la coherencia en toda la aplicación.

Historial de documentos

En la siguiente tabla se describen cambios importantes en esta guía. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una [fuente RSS](#).

Cambio	Descripción	Fecha
Reorganización del contenido	Actualizar la tabla de contenido y la organización del contenido para alinearla mejor con las demás AWS SDKs.	29 de marzo de 2025
Observabilidad	Se ha agregado información sobre la observabilidad de Ruby.	24 de enero de 2025
Actualizaciones generales	Se ha actualizado la versión mínima requerida de Ruby a v2.5. Se han actualizado los enlaces de recursos.	13 de noviembre de 2024
Índice y ejemplos guiados	Se han eliminado los ejemplos guiados para pasarlos al repositorio de ejemplos de código 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
Actualizaciones de las prácticas recomendadas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulta prácticas recomendadas de seguridad en IAM . Se	8 de mayo de 2023

ha actualizado el tutorial de introducción.

[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 ha actualizado la información de uso de la utilidad REPL para favorecer la claridad.

8 de agosto de 2022

[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

[SDK Metrics](#)

Se ha eliminado la información sobre la activación de SDK Metrics for Enterprise Support, que había 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.