Guida per gli sviluppatori

AWS SDK per Ruby

AWS SDK per Ruby: Guida per gli sviluppatori

Table of Contents

Cos'è l'AWSSDK for Ruby?

Benvenuto nella Guida per gli sviluppatori di AWS SDK for Ruby. L'AWSSDK for Ruby fornisce librerie di supporto per quasi tuttiServizi AWS, tra cui Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2) e Amazon DynamoDB.

L'AWSSDK for Ruby Developer Guide fornisce informazioni su come installare, configurare e utilizzare l'AWSSDK for Ruby per creare applicazioni Ruby che utilizzano. Servizi AWS

Inizia a usare l'AWSSDK for Ruby

Documentazione e risorse aggiuntive

Per ulteriori risorse per gli sviluppatori di AWS SDK for Ruby, consulta quanto segue:

- AWSGuida di riferimento agli SDK e agli strumenti: contiene impostazioni, funzionalità e altri concetti fondamentali comuni tra gli SDK AWS
- AWS SDK for RubyAPI Reference - Versione 3
- AWSArchivio di esempi di codice su GitHub
- RubyGems.org — L'ultima versione di SDK è modularizzata in gemme specifiche del servizio disponibili qui
  - Servizi supportati: elenca tutte le gemme supportate dall'AWSSDK for Ruby
- AWSFonte SDK for Ruby su: GitHub
  - Fonte e README
  - Registri delle modifiche sotto ogni gemma
  - Passare dalla v2 alla v3
  - Problemi
  - Note di base sull'aggiornamento
- Blog per sviluppatori
- Canale Gitter
- @awsforruby su Twitter

Implementazione nel cloud AWS

Puoi usare Servizi AWS ad esempio AWS Elastic BeanstalkAWS OpsWorks, e AWS CodeDeploy per distribuire la tua applicazione nel AWS Cloud. Per la distribuzione di applicazioni Ruby con Elastic Beanstalk, vedi Distribuzione di applicazioni Elastic Beanstalk in Ruby utilizzando EB CLI e Git nella Guida per gli sviluppatori. AWS Elastic Beanstalk Per distribuire un'applicazione Ruby on Rails conAWS OpsWorks, vedere Distribuzione di applicazioni Ruby on Rails su. AWS OpsWorks Per una panoramica dei servizi di AWS distribuzione, vedere Panoramica delle opzioni di distribuzione su. AWS

Manutenzione e supporto per le versioni principali dell'SDK

Per informazioni sulla manutenzione e sul supporto per le versioni principali dell'SDK e le relative dipendenze sottostanti, consulta quanto segue nella Guida di riferimento degli strumenti e degli SDK AWS:

- AWSPolitica di manutenzione degli SDK e degli strumenti
- AWSMatrice di Support delle versioni di SDK e strumenti

Inizia a usare l'AWSSDK for Ruby

Scopri come installare, configurare e utilizzare l'SDK per creare un'applicazione Ruby per accedere a una AWS risorsa in modo programmatico.

Argomenti

Autenticazione SDK conAWS

È necessario stabilire in che modo il codice si autentica conAWS quando si sviluppa conServizi AWS. È possibile configurare l'accesso programmatico alleAWS risorse in diversi modi a seconda dell'ambiente e dell'AWSaccesso disponibile.

Per scegliere il metodo di autenticazione e configurarlo per l'SDK, consulta Autenticazione e accesso nella Guida di riferimento agliAWS SDK e agli strumenti.

Consigliamo di configurare i nuovi utenti che si stanno sviluppando localmente e non hanno ricevuto un metodo di autenticazione dal datore di lavoroAWS IAM Identity Center. Questo metodo include l'installazione diAWS CLI per facilitare la configurazione e l'accesso regolare al portale diAWS accesso. Se si sceglie questo metodo, l'ambiente dovrebbe contenere i seguenti elementi dopo aver completato la procedura per l'autenticazione di IAM Identity Center nella Guida di riferimento agliAWS SDK e agli strumenti:

- IlAWS CLI, utilizzato per avviare una sessione del portale diAWS accesso prima di eseguire l'applicazione.
- Un AWS`config`file condiviso con un`[default]` profilo con un set di valori di configurazione a cui è possibile fare riferimento dall'SDK. Per trovare la posizione di questo file, consulta Posizione dei file condivisi nella Guida di riferimento agliAWS SDK e agli strumenti.
- Il`config` file condiviso imposta l'`region`impostazione. Questo imposta l'impostazione predefinitaRegione AWS utilizzata dall'SDK perAWS le richieste. Questa regione viene utilizzata per le richieste di servizi SDK che non sono specificate con una regione da utilizzare.

- L'SDK utilizza la configurazione del provider di token SSO del profilo per acquisire credenziali prima di inviare richieste aAWS. Il`sso_role_name` valore, che è un ruolo IAM collegato a un set di autorizzazioni IAM Identity Center, consente l'Servizi AWSaccesso all'utente nell'applicazione.

  Il seguente`config` file di esempio mostra un profilo predefinito impostato con la configurazione del provider di token SSO. L'`sso_session`impostazione del profilo si riferisce alla `sso-session`sezione denominata. L`sso-session` sezione contiene le impostazioni per avviare una sessione del portale diAWS accesso.

```
[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
```

L'AWSSDK for Ruby non richiede l'aggiunta di pacchetti aggiuntivi (come`SSO` and`SSOOIDC`) all'applicazione per utilizzare l'autenticazione IAM Identity Center.

Avvia una sessione del portale diAWS accesso

Prima di eseguire un'applicazione che accedeServizi AWS, è necessaria una sessione del portale diAWS accesso attiva affinché l'SDK utilizzi l'autenticazione IAM Identity Center per risolvere le credenziali. A seconda della durata della sessione configurata, l'accesso scadrà e l'SDK riscontrerà un errore di autenticazione. Per accedere al portale diAWS accesso, esegui il seguente comando inAWS CLI.

```
aws sso login
```

Se hai seguito le istruzioni e hai una configurazione predefinita del profilo, non è necessario chiamare il comando con un'`--profile`opzione. Se la configurazione del provider di token SSO utilizza un profilo denominato, il comando è`aws sso login --profile named-profile`.

Per verificare facoltativamente se hai già una sessione attiva, esegui ilAWS CLI comando seguente.

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

Se la sessione è attiva, la risposta a questo comando riporta l'account IAM Identity Center e il set di autorizzazioni configurati nel`config` file condiviso.

> **Note**
>
> Se hai già una sessione attiva del portale diAWS accesso ed eseguia`ws sso login`, non ti verrà richiesto di fornire le credenziali.
> La procedura di accesso potrebbe richiedere di consentire l'AWS CLIaccesso ai dati.
> PoichéAWS CLI è basato sull'SDK per Python, i messaggi di autorizzazione potrebbero contenere variazioni del`botocore` nome.

Informazioni dettagliate sull'autenticazione

Utenti umani, noti anche come identità umane, sono le persone, gli amministratori, gli sviluppatori, gli operatori e i consumatori delle tue applicazioni. Devono avere un'identità per accedere ai tuoi ambienti e alle tue applicazioni AWS. Gli utenti dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'umani dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'dell'

Usa credenziali temporanee per l'accessoAWS. È possibile utilizzare un provider di identità affinché gli utenti umani possano fornire l'accesso federato ad account AWS assumendo ruoli, che forniscono credenziali temporanee. Per una gestione centralizzata degli accessi, consigliamo di utilizzareAWS IAM Identity Center (IAM Identity Center) per gestire l'accesso ai tuoi account e le autorizzazioni all'interno di questi account. Per ulteriori alternative, consulta gli argomenti seguenti:

- Per ulteriori informazioni sulle best practice, consulta Best practice per la sicurezza in IAM in IAM.
- Per creareAWS credenziali a breve termine, consulta Credenziali di sicurezza temporanee nella Guida per l'utente IAM.
- Per informazioni su altri fornitori di credenzialiAWS SDK for Ruby, consulta Fornitori di credenziali standardizzati nella Guida di riferimento agliAWS SDK e agli strumenti.

Installa l'AWSSDK for Ruby

Questa sezione include i prerequisiti e le istruzioni di installazione per l'AWSSDK for Ruby.

Prerequisiti

Prima di utilizzare l'AWSSDK for Ruby, devi autenticarti con. AWS Nel caso di informazioni su come impostare l'autenticazione, consultaAutenticazione SDK conAWS.

Installazione dell'SDK

Puoi installare l'AWSSDK for Ruby come faresti con qualsiasi gemma Ruby. Le gemme sono disponibili su. RubyGems L'AWSSDK for Ruby è progettato per essere modulare ed è separato da. Servizio AWS L'installazione dell'intera `aws-sdk` gemma è grande e può richiedere più di un'ora.

Ti consigliamo di installare le gemme solo per il Servizi AWS tuo uso. Questi hanno lo stesso nome `aws-sdk-service_abbreviation` e l'elenco completo si trova nella tabella Servizi supportati del file AWS README SDK for Ruby. Ad esempio, la perla per l'interfacciamento con il servizio Amazon S3 è disponibile direttamente all'indirizzo. `aws-sdk-s3`

Gestore di versioni di Ruby

Invece di usare il sistema Ruby, consigliamo di utilizzare un gestore di versioni di Ruby come il seguente:

- RVM
- paffuto
- rbenv

Ad esempio, se si utilizza un sistema operativo Amazon Linux 2, è possibile utilizzare i seguenti comandi per aggiornare RVM, elencare le versioni di Ruby disponibili, quindi scegliere la versione che si desidera utilizzare per lo sviluppo con l'AWSSDK for Ruby. La versione minima richiesta di Ruby è la 2.3.

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

Raggruppatore

Se usi Bundler, i seguenti comandi installano l'AWSSDK for Ruby gem per Amazon S3:

1. Installa Bundler e crea: `Gemfile`

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

2. Apri il file creato `Gemfile` e aggiungi una `gem` riga per ogni gemma AWS di servizio che verrà utilizzata dal codice. Nel riportare l'esempio di Amazon S3, aggiungere la seguente riga alla fine del file:

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

3. Salva il Gemfile.
4. Installa le dipendenze specificate nel tuo`Gemfile`:

```
$ bundle install
```

Hello tutorial per l'AWSSDK for Ruby

Dai il benvenuto ad Amazon S3 utilizzando l'AWSSDK for Ruby. L'esempio seguente mostra un elenco dei tuoi bucket Amazon S3.

Scrivi il codice

Copia e incolla il codice seguente in un nuovo file sorgente. Assegnare un nome al file `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__
```

AWSSDK for Ruby è progettato per essere modulare ed è separato da. Servizio AWS Dopo l'installazione della gem, l'`require`istruzione nella parte superiore del file sorgente Ruby importa le classi e i metodi AWS SDK per il servizio Amazon S3. Per un elenco completo dei AWS service gem disponibili, consulta la tabella Supported Services del AWS file README SDK for Ruby.

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

Esecuzione del programma

Apri un prompt dei comandi per eseguire il programma Ruby. La sintassi tipica dei comandi per eseguire un programma Ruby è:

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

Questo codice di esempio non utilizza argomenti. Per eseguire questo codice, inserisci quanto segue nel prompt dei comandi:

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

Nota per gli utenti Windows

Quando utilizzi certificati SSL su Windows ed esegui il codice Ruby, potresti visualizzare un errore simile al seguente.

```
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'
...
```

Per risolvere questo problema, aggiungi la riga seguente al file sorgente di Ruby, da qualche parte prima della prima chiamata. AWS

```
Aws.use_bundled_cert!
```

Se stai usando solo la `aws-sdk-s3` gemma nel tuo programma Ruby e vuoi usare il certificato in bundle, devi aggiungere anche la gemma. `aws-sdk-core`

Passaggi successivi

Per testare molte altre operazioni di Amazon S3, consulta il AWSCode Examples Repository su. GitHub

Da usare AWS Cloud9 con l'AWSSDK for Ruby

AWS Cloud9è un ambiente di sviluppo integrato (IDE, Integrated Development Environment) basato su Web che contiene una serie di strumenti che consente di scrivere, gestire il debug e il rilascio di software nel cloud. Puoi usarlo AWS Cloud9 con l'AWSSDK for Ruby per scrivere ed eseguire il tuo codice Ruby utilizzando un browser. AWS Cloud9include strumenti come un editor di codice e un terminale. Poiché l'AWS Cloud9IDE è basato sul cloud, puoi lavorare ai tuoi progetti dal tuo ufficio, da casa o ovunque utilizzando una macchina connessa a Internet. Per informazioni generali suAWS Cloud9, vedere la Guida per AWS Cloud9 l'utente.

Segui queste istruzioni per eseguire la configurazione AWS Cloud9 con l'AWSSDK for Ruby:

- Fase 1: Account AWS configurare il AWS Cloud9
- Fase 2: configurare il tuo ambiente di AWS Cloud9 sviluppo
- Fase 3: configurare l'AWSSDK for Ruby
- Fase 4: Download del codice di esempio
- Fase 5: Esecuzione del codice

Fase 1: Account AWS configurare il AWS Cloud9

Per AWS Cloud9 utilizzarlo, accedi alla AWS Cloud9 console daAWS Management Console.

> **Note**
>
> Se si utilizza AWS IAM Identity Center per l'autenticazione, potrebbe essere necessario aggiungere l'autorizzazione richiesta di `iam:ListInstanceProfilesForRole` alla policy allegata dall'utente nella console IAM.

Per configurare un'entità IAM nel tuo AWS account per accedere AWS Cloud9 e accedere alla AWS Cloud9 console, consulta Team Setup AWS Cloud9 nella Guida per l'AWS Cloud9utente.

Fase 2: configurare il tuo ambiente di AWS Cloud9 sviluppo

Dopo aver effettuato l'accesso alla console di AWS Cloud9, utilizzala per creare un ambiente di sviluppo di AWS Cloud9. In seguito alla creazione dell'ambiente, AWS Cloud9 aprirà l'IDE per tale ambiente.

Per ulteriori informazioni, vedere Creazione di un ambiente AWS Cloud9 nella Guida per l'AWS Cloud9utente.

> **Note**
>
> Dal momento che stai creando l'ambiente nella console per la prima volta, ti consigliamo di scegliere l'opzione Create a new instance for environment (EC2) (Crea una nuova istanza per l'ambiente (EC2)). Questa opzione indica AWS Cloud9 di creare un ambiente, gestire

un'istanza di Amazon EC2 e quindi connettere la nuova istanza al nuovo ambiente. Si tratta del modo più rapido per iniziare a utilizzare AWS Cloud9.

Apri il terminale se non è già aperto nell'IDE. Nella barra del menu nell'IDE, scegli Window, New Terminal (Finestra, Nuovo terminale). Puoi usare la finestra del terminale per installare strumenti e creare le tue applicazioni.

Fase 3: configurare l'AWSSDK for Ruby

Dopo aver AWS Cloud9 aperto l'IDE per il tuo ambiente di sviluppo, usa la finestra del terminale per configurare l'AWSSDK for Ruby nel tuo ambiente.

Puoi installare l'AWSSDK for Ruby come faresti con qualsiasi gemma Ruby. Le gemme sono disponibili su. RubyGems L'AWSSDK for Ruby è progettato per essere modulare ed è separato da. Servizio AWS L'installazione dell'intera `aws-sdk` gemma è grande e può richiedere più di un'ora.

Ti consigliamo di installare le gemme solo per il Servizi AWS tuo uso. Questi hanno lo stesso nome `aws-sdk-`*`service_abbreviation`* e l'elenco completo si trova nella tabella Servizi supportati del file AWS README SDK for Ruby. Ad esempio, la perla per l'interfacciamento con il servizio Amazon S3 è disponibile direttamente all'indirizzo. `aws-sdk-s3`

Gestione delle versioni di Ruby

Invece di usare il sistema Ruby, consigliamo di utilizzare un gestore di versioni di Ruby come il seguente:

- RVM
- paffuto
- rbenv

Ad esempio, se si utilizza un sistema operativo Amazon Linux 2, è possibile utilizzare i seguenti comandi per aggiornare RVM, elencare le versioni di Ruby disponibili, quindi scegliere la versione che si desidera utilizzare per lo sviluppo con l'AWSSDK for Ruby. La versione minima richiesta di Ruby è la 2.3.

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

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

Raggruppatore

Se usi Bundler, i seguenti comandi installano l'AWSSDK for Ruby gem per Amazon S3:

1. Installa Bundler e crea: `Gemfile`

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

2. Apri il file creato `Gemfile` e aggiungi una `gem` riga per ogni gemma AWS di servizio che verrà utilizzata dal codice. Per seguire l'esempio di Amazon S3 alla fine del file alla fine del file alla fine del file alla fine del file alla fine del file alla fine del file alla fine del file alla fine del

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

3. Salvataggio del Gemfile.
4. Installa le dipendenze specificate nel tuo`Gemfile`:

   ```
   $ bundle install
   ```

Fase 4: Download del codice di esempio

Usa la finestra del terminale per scaricare il codice di esempio per l'AWSSDK for Ruby nell'ambiente di AWS Cloud9 sviluppo.

Per scaricare una copia di tutti gli esempi di codice utilizzati nella documentazione ufficiale dell'AWSSDK nella directory principale del tuo ambiente, esegui il seguente comando:

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

Gli esempi di codice per l'AWSSDK for Ruby si trovano `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby` nella directory, dove si `ENVIRONMENT_NAME` trova il nome del tuo ambiente di sviluppo.

Per continuare utilizzando un esempio di Amazon S3, ti consigliamo di iniziare con un esempio `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby/example_code/s3/bucket_list.rb` di codice. Usa la finestra del terminale per accedere alla `s3` directory ed elencare i file.

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

Per aprire il file inAWS Cloud9, puoi fare clic `bucket_list.rb` direttamente nella finestra del terminale.

Per ulteriore supporto nella comprensione degli esempi di codice, consulta AWSSDK for Ruby Code Examples.

Fase 5: Esecuzione del codice

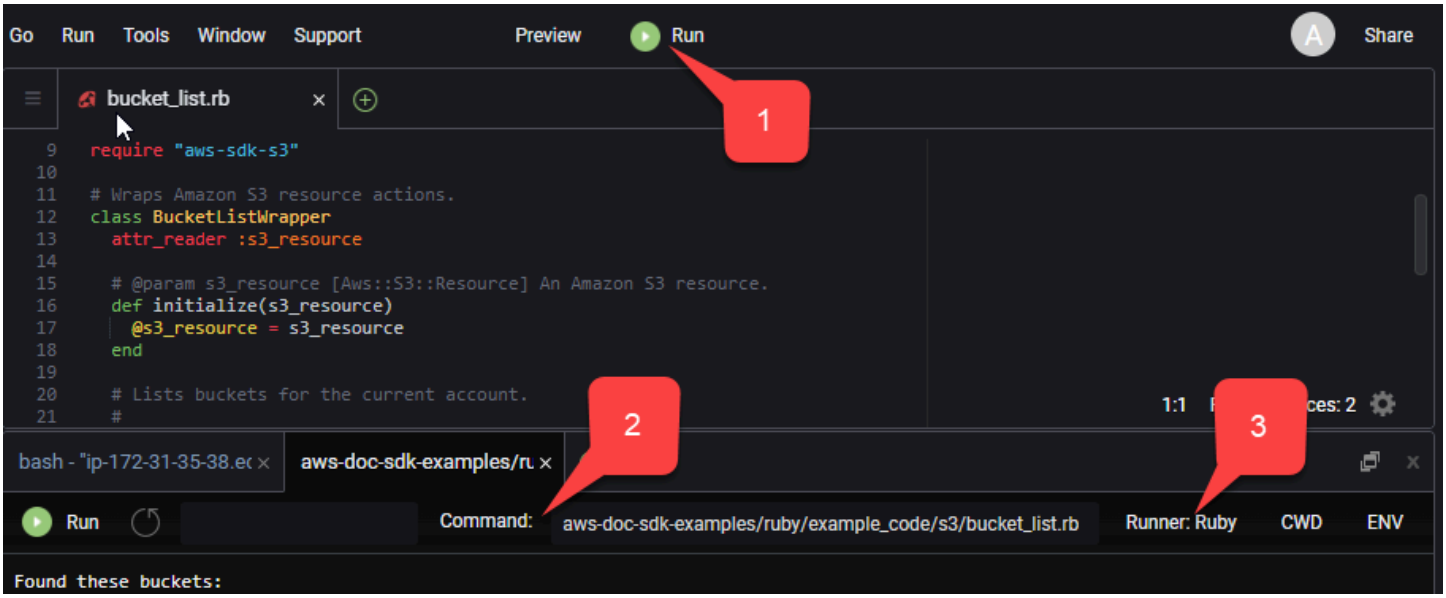
Per eseguire il codice nel tuo ambiente di AWS Cloud9 sviluppo, scegli il pulsante Esegui nella barra dei menu in alto. AWS Cloud9rileverà automaticamente l'estensione del `.rb` file e utilizzerà Ruby runner per eseguire il codice. Per ulteriori informazioni sull'esecuzione del codice inAWS Cloud9, consulta Esegui il tuo codice nella Guida per l'AWS Cloud9utente.

Nella schermata seguente, annotare le seguenti aree di base:

- 1: Esecuzione. Il pulsante Esegui si trova nella barra dei menu in alto. Si apre una nuova scheda per i risultati.

  > **Note**
  >
  > È anche possibile creare manualmente nuove configurazioni di esecuzione. Nella barra dei menu, selezionare Run (Esegui), Run Configurations (Configurazioni esecuzione), New Run Configuration (Nuova configurazione esecuzione).

- 2: Comando. AWS Cloud9compila la casella di testo Comando con il percorso e il nome del file che esegui. Se il codice prevede che vengano trasmessi parametri della riga di comando, questi possono essere aggiunti alla riga di comando nello stesso modo in cui si esegue il codice tramite una finestra di terminale.
- 3: Corridore. AWS Cloud9rileva che l'estensione del file è `.rb` e seleziona Ruby Runner per eseguire il codice.



Qualsiasi output generato dal codice in esecuzione viene visualizzato nella scheda.

Configurare l'AWSSDK per Ruby

Scopri come configurare l'AWSSDK per Ruby. È necessario stabilire in che modo il codice si autentica AWS durante lo sviluppo con. Servizi AWS È inoltre necessario impostare Regione AWS quello che si desidera utilizzare.

Catena di fornitori di credenziali

Tutti gli SDK dispongono di una serie di posizioni (o fonti) che controllano per ottenere credenziali valide da utilizzare per effettuare una richiesta a un. Servizio AWS Dopo aver trovato credenziali valide, la ricerca viene interrotta. Questa ricerca sistematica è denominata catena di fornitori di credenziali predefinita.

Per ogni fase della catena, esistono diversi modi per impostare i valori. L'impostazione dei valori direttamente nel codice ha sempre la precedenza, seguita dall'impostazione come variabili di ambiente e quindi nel AWS `config` file condiviso. Per ulteriori informazioni, consulta Precedenza delle impostazioni nella Guida di riferimento agli AWSSDK e agli strumenti.

La Guida di riferimento agli AWS SDK e agli strumenti contiene informazioni sulle impostazioni di configurazione SDK utilizzate da tutti gli AWS SDK e da. AWS CLI Per ulteriori informazioni su come configurare l'SDK tramite il AWS `config` file condiviso, consulta File di configurazione e credenziali condivisi. Per ulteriori informazioni su come configurare l'SDK tramite l'impostazione delle variabili di ambiente, consulta Supporto per le variabili di ambiente.

Con cui eseguire l'autenticazioneAWS, l'AWSSDK for Ruby controlla i provider di credenziali nell'ordine elencato nella tabella seguente.

Fornitore di credenziali per precedenza	AWSGuida di riferimento agli SDK e agli strumenti	AWS SDK for Ruby Documentazione di riferimento delle API

Credenziali statiche	AWSchiavi di accesso	`Aws::Credentials`
		
		`Aws::SharedCredentials`

Fornitore di credenziali per precedenza	AWSGuida di riferimento agli SDK e agli strumenti	AWS SDK for Ruby Documentazione di riferimento delle API
Token di identità Web da AWS Security Token Service (AWS STS)	Assumi il ruolo di fornitore di credenziali Utilizzando `role_arn` `role_session_name` , e `web_identity_token_file`	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center. In questa guida, vediAutenticazione SDK conAWS.	Provider di credenziali IAM Identity Center	`Aws::SSOCredentials`
Fornitore di entità affidabile (ad esempio`AWS_ROLE_ARN`). In questa guida, vediCreazione di un token di AWS STS accesso.	Assumi il ruolo di fornitore di credenziali Usando `role_arn` e `role_session_name`	`Aws::AssumeRoleCredentials`
Provider di credenziali di processo	Fornitore di credenziali di processo	`Aws::ProcessCredentials`
Credenziali Amazon Elastic Container Service (Amazon ECS)	Fornitore di credenziali per container	`Aws::ECSCredentials`
Credenziali del profilo dell'istanza Amazon Elastic Compute Cloud (Amazon EC2) (provider di credenziali IMDS)	Provider di credenziali IMDS	`Aws::InstanceProfileCredentials`

Se la variabile di `AWS_SDK_CONFIG_OPT_OUT` ambiente AWS SDK for Ruby è impostata, AWS `config` il file condiviso, in `~/.aws/config` genere at, non verrà analizzato per le credenziali.

Se avete seguito l'approccio consigliato per i nuovi utenti per iniziare, impostate AWS IAM Identity Center l'autenticazione durante l'argomento Guida Autenticazione SDK conAWS introduttiva. Altri metodi di autenticazione sono utili per diverse situazioni. Per evitare rischi per la sicurezza, consigliamo di utilizzare sempre credenziali a breve termine. Per altre procedure relative ai metodi di autenticazione, consulta Autenticazione e accesso nella Guida di riferimento agli AWS SDK e agli strumenti.

Creazione di un token di AWS STS accesso

Assumere un ruolo implica l'utilizzo di un set di credenziali di sicurezza temporanee che è possibile utilizzare per accedere a AWS risorse a cui normalmente non si ha accesso. Le credenziali temporanee sono costituite da un ID chiave di accesso, una chiave di accesso segreta e un token di sicurezza. È possibile utilizzare il `Aws::AssumeRoleCredentials`metodo per creare un token di accesso AWS Security Token Service (AWS STS).

L'esempio seguente utilizza un token di accesso per creare un oggetto client Amazon S3, dove si `linked::account::arn` trova l'Amazon Resource Name (ARN) del ruolo da assumere ed `session-name` è un identificatore per la sessione di ruolo assunta.

```
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)
```

Per ulteriori informazioni sull'impostazione `role_arn` o`role_session_name`, o su come impostarli utilizzando invece il AWS `config` file condiviso, consulta Assume role Credential Provider nella AWSSDK and Tools Reference Guide.

Impostazione di una regione

È necessario impostare una regione quando si utilizza la maggior parteServizi AWS. L'AWSSDK for Ruby cerca una regione nel seguente ordine:

1. Impostazione della regione in un client o in un oggetto risorsa
2. Impostazione della regione utilizzando `Aws.config`

3. Impostazione della regione utilizzando variabili di ambiente
4. Impostazione della regione utilizzando il `config` file condiviso

Per ulteriori informazioni sull'`region`impostazione, consulta la Guida Regione AWSdi riferimento agli AWS SDK e agli strumenti. Il resto di questa sezione descrive come impostare una regione, a partire dall'approccio più comune.

Impostazione della regione utilizzando il **`config`** file condiviso

Imposta la regione impostando la `region` variabile nel AWS `config` file condiviso. Per ulteriori informazioni sul `config` file condiviso, consulta File di configurazione e credenziali condivisi nella Guida di riferimento agli AWSSDK e agli strumenti.

Esempio di impostazione di questo valore nel file: `config`

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

Il `config` file condiviso non viene controllato se la variabile di ambiente `AWS_SDK_CONFIG_OPT_OUT` è impostata.

Impostazione della regione utilizzando le variabili di ambiente

Imposta la regione impostando la variabile di `AWS_REGION` ambiente.

Usa il `export` comando per impostare questa variabile su sistemi basati su Unix, come Linux o macOS. L'esempio seguente imposta la regione su. `us-west-2`

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

Per impostare questa variabile su Windows, utilizzate il `set` comando. L'esempio seguente imposta la regione su`us-west-2`.

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

Impostazione della regione con **`Aws.config`**

Imposta la regione aggiungendo un `region` valore all'`Aws.config`hash. L'esempio seguente aggiorna l'`Aws.config`hash per utilizzare la `us-west-1` regione.

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

Tutti i client o le risorse che crei successivamente sono associati a questa regione.

Impostazione della regione in un oggetto client o risorsa

Imposta la regione quando crei un AWS client o una risorsa. L'esempio seguente crea un oggetto risorsa Amazon S3 nella `us-west-1` regione. Scegli la regione corretta per le tue AWS risorse. Un oggetto client di servizio è immutabile, quindi è necessario creare un nuovo client per ogni servizio a cui si effettuano richieste e per effettuare richieste allo stesso servizio utilizzando una configurazione diversa.

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

Impostazione di un endpoint non standard

La regione viene utilizzata per costruire un endpoint SSL da utilizzare per le richieste. AWS Se devi utilizzare un endpoint non standard nella regione che hai selezionato, aggiungi una voce a. `endpoint` `Aws.config` In alternativa, imposta `endpoint:` quando crei un client di servizio o un oggetto risorsa. L'esempio seguente crea un oggetto risorsa Amazon S3 nell'`other_endpoint`endpoint.

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

Per utilizzare un endpoint di tua scelta per le richieste API e far sì che tale scelta venga mantenuta, consulta l'opzione di configurazione degli endpoint specifici del servizio nella Guida di riferimento agli SDK e agli strumenti. AWS

Usa l'AWSSDK for Ruby

Questa sezione fornisce informazioni sullo sviluppo di software con l'AWSSDK for Ruby, incluso come utilizzare alcune delle funzionalità avanzate dell'SDK.

La Guida di riferimento agli AWS SDK e agli strumenti contiene anche impostazioni, funzionalità e altri concetti fondamentali comuni a molti SDK. AWS

Argomenti

Usa l'utilità AWS SDK for Ruby REPL

La `aws-sdk` gemma include un'interfaccia interattiva a riga di comando Read-Eval-Print-Loop (REPL) in cui è possibile testare l'SDK for Ruby e vedere immediatamente i risultati. Le gemme SDK for Ruby sono disponibili all'indirizzo .org. RubyGems

Prerequisiti

- Installa l'AWSSDK for Ruby.
- `aws-v3.rb`Si trova nella gemma. `aws-sdk-resources` La `aws-sdk-resources` gemma è inclusa anche nella gemma principale `aws-sdk`.
- Avrai bisogno di una libreria xml, come la `rexml` gem.
- Sebbene il programma funzioni con Interactive Ruby Shell (`irb`), si consiglia di installare la `pry` gem, che fornisce un ambiente REPL più potente.

Configurazione del bundler

Se utilizzi Bundler, i seguenti aggiornamenti `Gemfile` riguarderanno i gemme prerequisiti:

1. Apri il file `Gemfile` che hai creato quando hai installato l'AWSSDK for Ruby. Aggiungere le seguenti righe al file:

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

2. Salva il Gemfile.
3. Installa le dipendenze specificate nel tuo: `Gemfile`

```
$ bundle install
```

Esecuzione di REPL

È possibile accedere al REPL eseguendolo `aws-v3.rb` dalla riga di comando.

```
aws-v3.rb
```

In alternativa, è possibile abilitare la registrazione via cavo HTTP impostando il flag verbose. La registrazione via cavo HTTP fornisce informazioni sulla comunicazione tra AWS SDK for Ruby e. AWS Nota, il flag verbose aggiunge anche un sovraccarico che può rallentare l'esecuzione del codice.

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

L'SDK for Ruby include classi client che forniscono interfacce a. Servizi AWS Ogni classe client supporta una particolare. Servizio AWS Nel REPL, ogni classe di servizio ha un helper che restituisce un nuovo oggetto client per interagire con quel servizio. Il nome dell'helper sarà il nome del servizio convertito in lettere minuscole. Ad esempio, i nomi degli oggetti `s3` helper di Amazon S3 e Amazon EC2 sono e, rispettivamente. `ec2` Per elencare i bucket Amazon S3 presenti nel tuo account, puoi `s3.list_buckets` accedere al prompt.

Puoi digitare `quit` nel prompt REPL per uscire.

Usa l'SDK con Ruby on Rails

Ruby on Rails fornisce un framework di sviluppo web che semplifica la creazione di siti Web con Ruby.

AWSfornisce la `aws-sdk-rails` gemma per consentire una facile integrazione con Rails. Puoi usareAWS Elastic Beanstalk, AWS OpsWorksAWS CodeDeploy, o AWSRails Provisioner per distribuire ed eseguire le tue applicazioni Rails nel cloud. AWS

Per informazioni sull'installazione e l'uso della `aws-sdk-rails` gem, consulta il GitHub repository https://github.com/aws/. aws-sdk-rails

Suggerimento per il debug: ottieni informazioni sulle tracce dei cavi da un client

È possibile ottenere informazioni sulla tracciabilità dei cavi da un AWS client impostando il valore `http_wire_trace` booleano. Le informazioni sulla tracciabilità dei cavi aiutano a differenziare le modifiche dei clienti, i problemi di servizio e gli errori degli utenti. Quando`true`, l'impostazione mostra cosa viene inviato via cavo. L'esempio seguente crea un client Amazon S3 con tracciamento dei cavi abilitato al momento della creazione del client.

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

Dato il codice seguente e l'argomento`bucket_name`, l'output visualizza un messaggio che indica se esiste un bucket con quel nome.

```
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
```

Se il bucket è simile a quello riportato di seguito. (I ritorni sono stati aggiunti alla `HEAD` riga per facilitarne la leggibilità.)

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

Puoi anche attivare il tracciamento dei cavi dopo la creazione del client.

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

Per ulteriori informazioni sui campi delle informazioni sulle tracce cablate riportate, consulta le intestazioni delle richieste obbligatorie di Transfer Family.

Risposte ed errori dei client Stub

Scopri come bloccare le risposte e gli errori dei client in un'applicazione AWS SDK for Ruby.

Stubbing delle risposte dei clienti

Quando si interrompe una risposta, l'AWSSDK for Ruby disattiva il traffico di rete e il client restituisce dati stub (o falsi). Se non si dispone di dati stub, il cliente restituisce:

- Elenchi come matrici vuote
- Mappe come hash vuoti
- Valori numerici come zero
- Date come now

L'esempio seguente restituisce nomi stub per l'elenco dei bucket 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
```

L'esecuzione di questo codice mostra quanto segue.

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

> **Note**
>
> Dopo aver fornito dati stub, i valori predefiniti non si applicano più agli attributi di istanza rimanenti. Ciò significa che nell'esempio precedente, l'attributo di istanza rimanente,`creation_date`, non è now `nil`.

L'AWSSDK for Ruby convalida i dati stub. Se si trasmettono dati del tipo sbagliato, viene generata un'`ArgumentError`eccezione. Ad esempio, se invece della precedente assegnazione a`bucket_data`, hai utilizzato quanto segue:

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

L'AWSSDK for Ruby solleva due eccezioni. `ArgumentError`

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

Stubbing degli errori del client

Puoi anche stub degli errori che l'AWSSDK for Ruby genera per metodi specifici. Viene visualizzato l'esempio seguente`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
```

Impaginazione

Alcune AWS chiamate forniscono risposte in pagine per limitare la quantità di dati restituiti con ciascuna risposta. Una pagina costituita da un massimo di 1.000 elementi.

Le risposte per pagina sono numerabili

Il modo più semplice per gestire i dati di risposta paginati consiste nell'utilizzare l'enumeratore integrato nell'oggetto di risposta, come illustrato nell'esempio seguente.

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

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

Ciò produce un oggetto di risposta per chiamata API effettuata ed enumera gli oggetti nel bucket denominato. L'SDK recupera pagine di dati aggiuntive per completare la richiesta.

Gestione manuale delle risposte suddivise in pagine

Per gestire autonomamente il paging, utilizza il `next_page?` metodo della risposta per verificare che ci siano più pagine da recuperare oppure utilizza il `last_page?` metodo per verificare che non ci siano altre pagine da recuperare.

Se ci sono più pagine, utilizzate il metodo `next_page` (notate che non c'è?) per recuperare la pagina successiva di risultati, come mostrato nell'esempio seguente.

```
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**
>
> Se chiami il `next_page` metodo e non ci sono più pagine da recuperare, l'SDK genera un'eccezione Aws:::PageableResponse:. LastPageError

Classi di dati paginate

I dati paginati nell'AWSSDK for Ruby sono gestiti dalla PageableResponse classe Aws::, inclusa in Seahorse: :Client: :Response per fornire l'accesso ai dati paginati.

Camerieri

I camerieri sono metodi di utilità che effettuano sondaggi per verificare che un determinato stato si verifichi su un cliente. I camerieri possono fallire dopo un certo numero di tentativi a un intervallo di polling definito per il client del servizio. Per un esempio di come viene utilizzato un cameriere, consulta il metodo create_table del client di crittografia Amazon DynamoDB nel Code Examples Repository. AWS

Richiamo di un cameriere

Per chiamare un cameriere, chiama un cliente `wait_until` del servizio. Nell'esempio seguente, un cameriere attende che l'istanza `i-12345678` sia in funzione prima di continuare.

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

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

Il primo parametro è il nome del cameriere, che è specifico del client del servizio e indica quale operazione è in attesa. Il secondo parametro è un hash di parametri che vengono passati al metodo client chiamato dal cameriere, che varia in base al nome del cameriere.

Per un elenco delle operazioni che possono essere attese e dei metodi client richiesti per ciascuna operazione, consulta la `waiter_names` documentazione `wait_until` sul campo del client che stai utilizzando.

Attendere i problemi di attesa

I camerieri possono fallire con una delle seguenti eccezioni.

Aws: :Camerieri: :Errori:: FailureStateError

Si è verificato uno stato di errore durante l'attesa.

Aws: :Camerieri: :Errori:: NoSuchWaiterError

Il nome del cameriere specificato non è definito per il cliente utilizzato.

Aws: :Camerieri: :Errori:: TooManyAttemptsError

Il numero di tentativi ha superato il valore del cameriere. `max_attempts`

Aws: :Camerieri: :Errori:: UnexpectedError

Si è verificato un errore imprevisto durante l'attesa.

Aws: :Camerieri: :Errori:: WaiterFailed

Uno degli stati di attesa è stato superato o si è verificato un altro errore durante l'attesa.

Tutti questi errori, `NoSuchWaiterError` tranne, si basano su. `WaiterFailed` Per catch gli errori in un cameriere utilizza`WaiterFailed`, come illustrato nell'esempio seguente.

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

Configurazione di un cameriere

Ogni cameriere ha un intervallo di polling predefinito e un numero massimo di tentativi da effettuare prima di restituire il controllo al programma. Per impostare questi valori, usa i `delay:` parametri `max_attempts` and nella `wait_until` chiamata. L'esempio seguente attende fino a 25 secondi, effettuando un sondaggio ogni cinque secondi.

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

Per disattivare gli errori di attesa, impostate il valore di uno di questi parametri su`nil`.

Estendi un cameriere

Per modificare il comportamento dei camerieri, puoi registrare i callback che vengono attivati prima di ogni tentativo di polling e prima di attendere.

L'esempio seguente implementa un arretramento esponenziale in un cameriere raddoppiando il tempo di attesa ad ogni tentativo.

```
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
```

L'esempio seguente disabilita il numero massimo di tentativi e attende invece un'ora (3600 secondi) prima di fallire.

```
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
```

Specificare il comportamento dei nuovi tentativi del client

Per impostazione predefinita, l'AWSSDK for Ruby esegue fino a tre tentativi, con 15 secondi tra un tentativo e l'altro, per un totale di quattro tentativi. Pertanto, il timeout di un'operazione potrebbe richiedere fino a 60 secondi.

L'esempio seguente crea un client Amazon S3 nella regione `us-west-2` e specifica di attendere cinque secondi tra due tentativi per ogni operazione del client. Pertanto, il timeout delle operazioni del client Amazon S3 potrebbe richiedere fino a 15 secondi.

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

Questo esempio mostra come modificare i parametri di riprova direttamente all'interno del codice. Tuttavia, puoi anche utilizzare le variabili di ambiente o il AWS `config` file condiviso per impostarle per la tua applicazione. Per ulteriori informazioni su queste impostazioni, consulta Retry behavior nella Guida di riferimento agli AWS SDK e agli strumenti. Qualsiasi impostazione esplicita impostata nel codice o su un client di servizio stesso ha la precedenza su quelle impostate nelle variabili di ambiente o nel file condiviso. `config`

Esegui la migrazione dalla versione 1 o 2 alla versione 3 dell'AWSSDK for Ruby

Lo scopo di questo argomento è aiutarti a migrare dalla versione 1 o 2 dell'AWSSDK for Ruby alla versione 3.

ide-by-sideUtilizzo S

Non è necessario sostituire la versione 1 o 2 dell'AWSSDK for Ruby con la versione 3. Puoi usarli insieme nella stessa applicazione. Leggi questo post sul blog per ulteriori informazioni.

Segue un rapido esempio.

```
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
```

Non è necessario riscrivere il codice della versione 1 o 2 funzionante esistente per iniziare a utilizzare l'SDK della versione 3. Una strategia di migrazione valida consiste nello scrivere nuovo codice solo sull'SDK versione 3.

Differenze generali

La versione 3 differisce dalla versione 2 in un modo importante.

- Ogni servizio è disponibile come gemma separata.

La versione 2 differisce dalla versione 1 in diversi modi importanti.

- Namespace principale diverso: `Aws` versus. `AWS` Ciò consente side-by-side l'utilizzo.
- `Aws.config`— Ora un hashish Ruby alla vaniglia, invece di un metodo.
- Opzioni di costruzione rigorose: quando si costruisce un oggetto client o risorsa nell'SDK versione 1, le opzioni di costruzione sconosciute vengono ignorate. Nella versione 2, le opzioni del costruttore sconosciute attivano un`ArgumentError`. Ad esempio:

```
# 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
```

Differenze tra i clienti

Non ci sono differenze tra le classi client nella versione 2 e nella versione 3.

Tra la versione 1 e la versione 2, le classi client presentano il minor numero di differenze esterne. Molti clienti di servizio disporranno di interfacce compatibili dopo la costruzione del cliente. Alcune note differenze:

- `Aws::S3::Client`- La classe client Amazon S3 versione 1 è stata codificata a mano. La versione 2 è generata da un modello di servizio. I nomi e gli input dei metodi sono molto diversi nella versione 2.
- `Aws::EC2::Client`- La versione 2 utilizza nomi plurali per gli elenchi di output, la versione 1 utilizza il suffisso`_set`. Ad esempio:

```
# 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 versione 2 utilizza risposte strutturate, mentre la versione 1 utilizza hash Ruby vanilla.
- Rinomina delle classi di servizio: la versione 2 utilizza un nome diverso per più servizi:
  - `AWS::SimpleWorkflow`è diventato `Aws::SWF`
  - `AWS::ELB`è diventato `Aws::ElasticLoadBalancing`
  - `AWS::SimpleEmailService`è diventato `Aws::SES`

- Opzioni di configurazione del client: alcune delle opzioni di configurazione della versione 1 sono state rinominate nella versione 2. Altri vengono rimossi o sostituiti. Ecco le modifiche principali:
  - `:use_ssl` è stato rimosso. La versione 2 utilizza SSL ovunque. Per disabilitare SSL è necessario configurare e `:endpoint` utilizzare`http://`.
  - `:ssl_ca_file`è ora `:ssl_ca_bundle`
  - `:ssl_ca_path`è ora `:ssl_ca_directory`
  - Aggiunto `:ssl_ca_store`.
  - `:endpoint`ora deve essere un URI HTTP o HTTPS completo anziché un nome host.
  - `:*_port`Opzioni rimosse per ogni servizio, ora sostituite da`:endpoint`.
  - `:user_agent_prefix`è ora `:user_agent_suffix`

Differenze tra le risorse

Non ci sono differenze tra le interfacce delle risorse nella versione 2 e nella versione 3.

Esistono differenze significative tra le interfacce delle risorse nella versione 1 e nella versione 2. La versione 1 è stata interamente codificata a mano, mentre nella versione 2 le interfacce di risorse sono generate da un modello. Le interfacce delle risorse della versione 2 sono significativamente più coerenti. Alcune delle differenze sistemiche includono:

- Classe di risorse separata: nella versione 2, il nome del servizio è un modulo, non una classe. In questo modulo, è l'interfaccia delle risorse:

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

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

- Risorse di riferimento: l'SDK versione 2 separa le raccolte e i singoli generatori di risorse in due metodi diversi:

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

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

- Operazioni in batch: nella versione 1, tutte le operazioni in batch erano utilità codificate a mano. Nella versione 2, molte operazioni batch sono operazioni di batch generate automaticamente tramite l'API. Le interfacce batch della versione 2 sono molto diverse dalla versione 1.

Lavora con Servizi AWS l'AWSSDK for Ruby

Le seguenti sezioni contengono discussioni ed esempi che mostrano come utilizzare l'AWSSDK for Ruby con cui lavorare. Servizi AWS

Se non conosci l'AWSSDK for Ruby, ti consigliamo di leggere prima Nozioni di base l'argomento.

- Esempi di codice con guida— Fornisce esempi guidati per diversi argomenti. Servizi AWS
- Esempi di codice— Fornisce un elenco completo degli esempi di servizi disponibili (ma senza ulteriori indicazioni oltre al codice).

Il codice sorgente di tutti questi esempi può essere scaricato nel AWSCode Examples Repository su GitHub. Per proporre un nuovo esempio di codice per il team della documentazione AWS che consideri la produzione, crea una nuova richiesta. Il team sta cercando di produrre esempi di codice che coprano scenari e casi d'uso più ampi rispetto ai semplici frammenti di codice che coprono solo le singole chiamate API. Per istruzioni, consultate la sezione Proporre nuovi esempi di codice nel file Readme on. GitHub

Esempi di codice con linee guida per l'AWSSDK for Ruby

Questa sezione fornisce esempi che è possibile utilizzare per accedere Servizi AWS utilizzando l'AWSSDK for Ruby.

Trovate il codice sorgente per questi e altri esempi nel AWSCode Examples Repository su. GitHub

Argomenti

- CloudTrail Esempi di utilizzo dell'AWSSDK for Ruby
- CloudWatch Esempi di Amazon che utilizzano l'AWSSDK for Ruby
- CodeBuild Esempi di utilizzo dell'AWSSDK for Ruby
- Esempi di Amazon EC2 che utilizzano l'AWSSDK for Ruby
- AWS Elastic BeanstalkEsempi di utilizzo dell'AWSSDK for Ruby
- AWS Identity and Access Management(IAM) Esempi di utilizzo dell'AWSSDK for Ruby
- AWS Key Management ServiceEsempi di utilizzo dell'AWSSDK for Ruby
- AWS LambdaEsempi di utilizzo dell'AWSSDK for Ruby
- Esempi di Amazon Polly con l'AWSSDK for Ruby

- Esempi di Amazon RDS con l'AWSSDK for Ruby
- Esempi di Amazon SES con l'AWSSDK for Ruby
- Esempi di Amazon SNS con l'AWSSDK for Ruby
- Esempi di Amazon SQS con l'AWSSDK for Ruby
- Esempi di Amazon WorkDocs

CloudTrail Esempi di utilizzo dell'AWSSDK for Ruby

CloudTrail è uno Servizio AWS strumento che puoi utilizzare per monitorare le tue AWS implementazioni nel cloud ottenendo una cronologia delle chiamate AWS API per il tuo account. È possibile utilizzare i seguenti esempi di codice AWS SDK for Ruby per accedere. AWS CloudTrail Per ulteriori informazioni su AWS CloudTrail; CloudTrail, consulta la documentazione.

Argomenti

- Elenco dei CloudTrail percorsi
- Creazione di un CloudTrail percorso
- Elenco degli eventi del CloudTrail percorso
- Eliminazione di un percorso CloudTrail

Elenco dei CloudTrail percorsi

Questo esempio utilizza il metodo describe_trails per elencare i nomi dei percorsi e il bucket in cui vengono CloudTrail memorizzate le informazioni nella regione. CloudTrail `us-west-2`

Scegliete `Copy` di salvare il codice localmente.

Crea il file describe_trails.rb con il codice seguente.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
```

```
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

resp = client.describe_trails({})

puts
puts "Found #{resp.trail_list.count} trail(s) in us-west-2:"
puts

resp.trail_list.each do |trail|
  puts 'Name:           ' + trail.name
  puts 'S3 bucket name: ' + trail.s3_bucket_name
  puts
end
```

Vedi l'esempio completo su. GitHub

Creazione di un CloudTrail percorso

Questo esempio utilizza il metodo create_trail per creare un CloudTrail trail nella regione. `us-west-2` Richiede due input, il nome del percorso e il nome del bucket in cui vengono memorizzate le informazioni. CloudTrail Se il bucket non ha la politica corretta, includi il flag -p per allegare la politica corretta al bucket.

Scegli di `Copy` salvare il codice localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'
```

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

# Attach IAM policy to bucket
def add_policy(bucket)
  # Get account ID using STS
  sts_client = Aws::STS::Client.new(region: 'us-west-2')
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to S3 bucket
  s3_client = Aws::S3::Client.new(region: 'us-west-2')

  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com',
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => 'arn:aws:s3:::' + bucket,
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com',
          },
          'Action' => 's3:PutObject',
          'Resource' => 'arn:aws:s3:::' + bucket + '/AWSLogs/' + account_id + '/*',
          'Condition' => {
            'StringEquals' => {
              's3:x-amz-acl' => 'bucket-owner-full-control',
            },
          },
        },
      ]
    }.to_json

    s3_client.put_bucket_policy(
```

```
      bucket: bucket,
      policy: policy
    )

    puts 'Successfully added policy to bucket ' + bucket
  rescue StandardError => err
    puts 'Got error trying to add policy to bucket ' + bucket + ':'
    puts err
    exit 1
  end
end

# main
name = ''
bucket = ''
attach_policy = false

i = 0

while i < ARGV.length
  case ARGV[i]
    when '-b'
      i += 1
      bucket = ARGV[i]

    when '-p'
      attach_policy = true

    else
      name = ARGV[i]
  end

  i += 1
end

if name == '' || bucket == ''
  puts 'You must supply a trail name and bucket name'
  puts USAGE
  exit 1
end

if attach_policy
  add_policy(bucket)
end
```

```
# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

begin
  client.create_trail({
    name: name, # required
    s3_bucket_name: bucket, # required
  })

  puts 'Successfully created CloudTrail ' + name + ' in us-west-2'
rescue StandardError => err
  puts 'Got error trying to create trail ' + name + ':'
  puts err
  exit 1
end
```

Guarda l'esempio completo su GitHub.

Elenco degli eventi del CloudTrail percorso

Questo esempio utilizza il metodo lookup_events per elencare gli eventi del CloudTrail trail nella regione. `us-west-2`

Scegliete `Copy` di salvare il codice localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

def show_event(event)
  puts 'Event name:   ' + event.event_name
  puts 'Event ID:     ' + event.event_id
```

```
  puts "Event time:   #{event.event_time}"
  puts 'User name:    ' + event.username

  puts 'Resources:'

  event.resources.each do |r|
    puts '  Name:       ' + r.resource_name
    puts '  Type:       ' + r.resource_type
    puts ''
  end
end

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

resp = client.lookup_events()

puts
puts "Found #{resp.events.count} events in us-west-2:"
puts

resp.events.each do |e|
  show_event(e)
end
```

Guarda l'esempio completo su GitHub.

Eliminazione di un percorso CloudTrail

Questo esempio utilizza il metodo delete_trail per eliminare una CloudTrail traccia nella regione. `us-west-2` Richiede un input, il nome del percorso.

Scegli `Copy` di salvare il codice localmente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
```

```
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-cloudtrail'  # v2: require 'aws-sdk'

if ARGV.length != 1
  puts 'You must supply the name of the trail to delete'
  exit 1
end

name = ARGV[0]

# Create client in us-west-2
client = Aws::CloudTrail::Client.new(region: 'us-west-2')

begin
  client.delete_trail({
    name: name, # required
  })

  puts 'Successfully deleted CloudTrail ' + name + ' in us-west-2'
rescue StandardError => err
  puts 'Got error trying to delete trail ' + name + ':'
  puts err
  exit 1
end
```

Guarda l'esempio completo su GitHub.

CloudWatch Esempi di Amazon che utilizzano l'AWSSDK for Ruby

Amazon CloudWatch (CloudWatch) è un servizio di monitoraggio per le risorse AWS cloud e le applicazioni su cui esegui. AWS È possibile utilizzare i seguenti esempi per accedere CloudWatch utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni in merito CloudWatch, consulta la documentazione di Amazon CloudWatch .

Argomenti

- Ottenere informazioni su Amazon CloudWatch Alarms
- Creazione di un Amazon Alarm CloudWatch
- Abilitazione e disabilitazione di Amazon CloudWatch Alarm Actions
- Ottenere informazioni sulle metriche personalizzate per Amazon CloudWatch

- Invio di eventi ad Amazon Events CloudWatch

Ottenere informazioni su Amazon CloudWatch Alarms

Il seguente esempio di codice mostra informazioni sugli allarmi metrici disponibili in Amazon. CloudWatch

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

require 'aws-sdk-cloudwatch'

# Displays information about available metric alarms in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts 'Name:           ' + alarm.alarm_name
      puts 'State value:    ' + alarm.state_value
      puts 'State reason:   ' + alarm.state_reason
      puts 'Metric:         ' + alarm.metric_name
      puts 'Namespace:      ' + alarm.namespace
      puts 'Statistic:      ' + alarm.statistic
      puts 'Period:         ' + alarm.period.to_s
      puts 'Unit:           ' + alarm.unit.to_s
      puts 'Eval. periods:  ' + alarm.evaluation_periods.to_s
      puts 'Threshold:      ' + alarm.threshold.to_s
      puts 'Comp. operator: ' + alarm.comparison_operator

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts '  ' + a
        end
      end
```

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

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

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts '  Name: ' + d.name + ', Value: ' + d.value
        end
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Full example call:
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__
```

Creazione di un Amazon Alarm CloudWatch

Il seguente esempio di codice crea un nuovo CloudWatch avviso (o aggiorna un avviso esistente, se esiste già un avviso con il nome specificato).

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

require 'aws-sdk-cloudwatch'

# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
```

```
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
```

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

# Full example call:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: 'BucketName',
      value: 'doc-example-bucket'
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  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
end

run_me if $PROGRAM_NAME == __FILE__
```

Abilitazione e disabilitazione di Amazon CloudWatch Alarm Actions

Il seguente esempio di codice:

1. Crea e attiva un nuovo CloudWatch avviso (o aggiorna un avviso esistente, se esiste già un avviso con il nome specificato).
2. Disattiva l'allarme nuovo o esistente. Per riattivare l'allarme, chiama`enable_alarm_actions`.

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

# The following code example shows how to:
# 1. Create or update an Amazon CloudWatch alarm.
# 2. Disable all actions for an alarm.

require 'aws-sdk-cloudwatch'
```

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
```

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

# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
```

```
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Full example call:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: 'BucketName',
      value: 'doc-example-bucket'
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  region = 'us-east-1'
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Ottenere informazioni sulle metriche personalizzate per Amazon CloudWatch

Il seguente esempio di codice:

1. Aggiunge punti dati a una metrica personalizzata in. CloudWatch
2. Visualizza un elenco di metriche disponibili per uno spazio dei nomi delle metriche in. CloudWatch

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

```
# The following example shows how to:
# 1. Add a datapoint to a metric in Amazon CloudWatch.
# 2. List available metrics for a metric namespace in Amazon CloudWatch.

require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
```

```
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end

# 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

# Full example call:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  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__
```

Invio di eventi ad Amazon Events CloudWatch

Il seguente esempio di codice mostra come creare e attivare una regola in Amazon CloudWatch Events. Questa regola invia una notifica all'argomento specificato in Amazon Simple Notification Service (Amazon SNS) ogni volta che un'istanza disponibile in Amazon Elastic Compute Cloud (Amazon EC2) Elastic Notification Service (Amazon SNS) passa allo stato di esecuzione. Inoltre, le informazioni relative agli eventi vengono registrate in un gruppo di log in Events. CloudWatch

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

# The following code example shows how to create and trigger a rule in
# Amazon CloudWatch Events. This rule sends a notification to the specified
# topic in Amazon Simple Notification Service (Amazon SNS) whenever an
# available instance in Amazon Elastic Compute Cloud (Amazon EC2) changes
# to a running state. Also, related event information is logged to a log group
# in Amazon CloudWatch Logs.
#
# This code example works with the following AWS resources through the
# following functions:
#
# - A rule in Amazon CloudWatch Events. See the rule_exists?, rule_found?,
#   create_rule, and display_rule_activity functions.
# - A role in AWS Identity and Access Management (IAM) to allow the rule
#   to work with Amazon CloudWatch Events. See role_exists?, role_found?,
#   and create_role.
# - An Amazon EC2 instance, which triggers the rule whenever it is restarted.
#   See instance_restarted?.
# - A topic and topic subscription in Amazon SNS for the rule to send event
#   notifications to. See topic_exists?, topic_found?, and create_topic.
# - A log group in Amazon CloudWatch Logs to capture related event information.
#   See log_group_exists?, log_group_created?, log_event, and display_log_data.
#
# This code example requires the following AWS resources to exist in advance:
#
```

```
# - An Amazon EC2 instance to restart, which triggers the rule.
#
# The run_me function toward the end of this code example calls the
# preceding functions in the correct order.

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'

# Checks whether the specified Amazon Simple Notification Service
# (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 Amazon Resource Name (ARN) of the
#   topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end

# Checks whether the specified topic exists among those available to the
# caller in Amazon Simple Notification Service (Amazon SNS).
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The Amazon Resource Name (ARN) of the
#   topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
```

```
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts 'Topic found.'
          return true
        end
      end
    end
  end
  puts 'Topic not found.'
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end

# Creates a topic in Amazon Simple Notification Service (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 Amazon Resource Name (ARN) of the topic that
#   was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
```

```
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return 'Error'
end

# 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 Amazon Resource Name (ARN) of the
#   role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

```
# 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 Amazon Resource Name (ARN) of the
#   role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts 'Role found.'
          return true
        end
      end
    end
  end
  puts 'Role not found.'
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end

# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon CloudWatch Events 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 Amazon Resource Name (ARN) of the role that
#   was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
```

```
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  return 'Error'
end

# Checks whether the specified AWS CloudWatch Events rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end

# Checks whether the specified rule exists among those available to the
# caller in AWS CloudWatch Events.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized AWS CloudWatch Events client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
```

```
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts 'Rule found.'
          return true
        end
      end
    end
  end
  puts 'Rule not found.'
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end

# Creates a rule in AWS CloudWatch Events.
# This rule is triggered whenever an available instance in
# Amazon Elastic Compute Cloud (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 Simple Notification Service (Amazon SNS).
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized AWS CloudWatch Events 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 Elastic Compute Cloud (Amazon EC2) must change to, to
```

```
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
```

```
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  return false
end

# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
```

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

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

# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
```

```
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

  response = cloudwatchlogs_client.put_log_events(event)
```

```
  puts 'Message logged.'
  return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end

# Restarts an Amazon Elastic Compute Cloud (Amazon EC2) instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''
```

```
  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

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

# Displays information about activity for a rule in Amazon CloudWatch Events.
#
# Prerequisites:
#
# - A rule in Amazon CloudWatch Events.
```

```
#
# @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

# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
```

```
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end

# 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 CloudWatch Events 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 CloudWatch Events 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
```

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

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

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

```
    exit 1
  end
end

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

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

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts 'Could not create the Amazon CloudWatch Logs log group ' \
    'correctly. Program stopped.'
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
```

```
  end

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

CodeBuild Esempi di utilizzo dell'AWSSDK for Ruby

CodeBuild è un servizio di compilazione completamente gestito che compila il codice sorgente, esegue test e produce pacchetti software pronti per la distribuzione. È possibile utilizzare i seguenti esempi di codice AWS SDK for Ruby per accedere. AWS CodeBuild Per ulteriori informazioni in merito CodeBuild, consulta la AWS CodeBuild documentazione.

Argomenti

- Ottenere informazioni su tutti i AWS CodeBuild progetti
- Costruire un AWS CodeBuild progetto
- Elenco delle build AWS CodeBuild del progetto

Ottenere informazioni su tutti i AWS CodeBuild progetti

L'esempio seguente elenca i nomi di un massimo di 100 progetti. AWS CodeBuild

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

resp = client.list_projects({
  sort_by: 'NAME', # accepts NAME, CREATED_TIME, LAST_MODIFIED_TIME
  sort_order: 'ASCENDING' # accepts ASCENDING, DESCENDING
})

resp.projects.each { |p| puts p }

puts
```

Scegliete di Copy salvare il codice localmente. Guarda l'esempio completo su GitHub.

Costruire un AWS CodeBuild progetto

L'esempio seguente crea il AWS CodeBuild progetto specificato nella riga di comando. Se non viene fornito alcun argomento della riga di comando, viene generato un errore e viene chiuso.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

project_name = ''

if ARGV.length != 1
  puts 'You must supply the name of the project to build'
  exit 1
else
  project_name = ARGV[0]
end

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

begin
  client.start_build(project_name: project_name)
  puts 'Building project ' + project_name
rescue StandardError => ex
  puts 'Error building project: ' + ex.message
end
```

Scegliete Copy di salvare il codice localmente. Guarda l'esempio completo su GitHub.

Elenco delle build AWS CodeBuild del progetto

L'esempio seguente mostra le informazioni sulle build del AWS CodeBuild progetto. Queste informazioni includono il nome del progetto, la data di inizio della compilazione e la durata di ciascuna fase della compilazione, in secondi.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
```

```
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-codebuild'  # v2: require 'aws-sdk'

client = Aws::CodeBuild::Client.new(region: 'us-west-2')

build_list = client.list_builds({sort_order: 'ASCENDING', })

builds = client.batch_get_builds({ids: build_list.ids})

builds.builds.each do |build|
  puts 'Project:    ' + build.project_name
  puts 'Phase:      ' + build.current_phase
  puts 'Status:     ' + build.build_status
end
```

Scegliete di Copy salvare il codice localmente. Guarda l'esempio completo su GitHub.

Esempi di Amazon EC2 che utilizzano l'AWSSDK for Ruby

Amazon Elastic Compute Cloud (Amazon EC2) è un servizio Web che fornisce una capacità di calcolo ridimensionabile, letteralmente server nei data center di Amazon, che puoi utilizzare per creare e ospitare i tuoi sistemi software. Puoi utilizzare i seguenti esempi per accedere ad Amazon EC2 utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon EC2, consulta la documentazione di Amazon EC2.

Argomenti

- Creazione di un VPC Amazon EC2
- Creazione di un Internet Gateway e collegamento a un VPC in Amazon EC2
- Creazione di una sottorete pubblica per Amazon EC2
- Creazione di una tabella di routing Amazon EC2 e associazione a una sottorete
- Utilizzo di indirizzi IP elastici in Amazon EC2
- Creazione di un gruppo di sicurezza Amazon EC2
- Lavorare con i gruppi di sicurezza di Amazon EC2

- Utilizzo delle coppie di chiavi in Amazon EC2
- Ottenere informazioni su tutte le istanze Amazon EC2
- Ottenere informazioni su tutte le istanze Amazon EC2 con un valore di tag specifico
- Ottenere informazioni su un'istanza Amazon EC2 specifica
- Creazione di un'istanza Amazon EC2
- Arresto di un'istanza Amazon EC2
- Avvio di un'istanza Amazon EC2
- Riavvio di un'istanza Amazon EC2
- Gestione delle istanze Amazon EC2
- Terminazione di un'istanza Amazon EC2
- Ottenere informazioni su regioni e zone di disponibilità per Amazon EC2

Creazione di un VPC Amazon EC2

Il seguente esempio di codice crea un cloud privato virtuale (VPC) in Amazon Virtual Private Cloud (Amazon VPC) e quindi contrassegna il VPC.

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

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-east-1'),
#     '10.0.0.0/24',
#     'my-key',
```

```
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return false
end

# Full example call:
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'
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-east-1'
    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'
    region = 'us-east-1'
  # 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__
```

Creazione di un Internet Gateway e collegamento a un VPC in Amazon EC2

Il seguente esempio di codice crea un gateway Internet e quindi lo collega a un cloud privato virtuale (VPC) in Amazon Virtual Private Cloud (Amazon VPC).

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

require 'aws-sdk-ec2'

# Creates an internet gateway and then attaches it to a virtual private cloud
# (VPC) in Amazon Virtual Private Cloud (Amazon VPC).
#
# 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 to attach the internet gateway.
```

```
# @param tag_key [String] The key of the tag to attach to the internet gateway.
# @param tag_value [String] The value of the tag to attach to the
#   internet gateway.
# @return [Boolean] true if the internet gateway was created and attached;
#   otherwise, false.
# @example
#   exit 1 unless internet_gateway_created_and_attached?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'vpc-6713dfEX'
#   )
def internet_gateway_created_and_attached?(
  ec2_resource,
  vpc_id,
  tag_key,
  tag_value
)
  igw = ec2_resource.create_internet_gateway
  puts "The internet gateway's ID is '#{igw.id}'."
  igw.attach_to_vpc(vpc_id: vpc_id)
  igw.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  return true
rescue StandardError => e
  puts "Error creating or attaching internet gateway: #{e.message}"
  puts 'If the internet gateway was created but not attached, you should ' \
    'clean up by deleting the internet gateway.'
  return false
end

# Full example call:
def run_me
  vpc_id = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
```

```
      'VPC_ID TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
      'vpc-6713dfEX my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

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

  if internet_gateway_created_and_attached?(
    ec2_resource,
    vpc_id,
    tag_key,
    tag_value
  )
    puts "Created and attached internet gateway to VPC '#{vpc_id}'."
  else
    puts "Could not create or attach internet gateway to VPC '#{vpc_id}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creazione di una sottorete pubblica per Amazon EC2

Il seguente esempio di codice crea una sottorete all'interno di un cloud privato virtuale (VPC) in Amazon Virtual Private Cloud (Amazon VPC) e quindi contrassegna la sottorete.

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

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-east-1'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-east-1a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
```

```
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return false
end

# Full example call:
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'
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-east-1a my-key my-value us-east-1'
    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-east-1a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # 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__
```

Creazione di una tabella di routing Amazon EC2 e associazione a una sottorete

Il seguente esempio di codice crea una tabella di routing in Amazon Virtual Private Cloud (Amazon VPC) e quindi associa la tabella di route a una sottorete in Amazon VPC.

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

require 'aws-sdk-ec2'

# Creates a route table in Amazon Virtual Private Cloud (Amazon VPC)
# and then associates the route table with a subnet in Amazon VPC.
#
# 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-east-1'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
```

```
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  return true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  return false
end

# Full example call:
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'
    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-east-1'
    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'
    region = 'us-east-1'
  # 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__
```

Utilizzo di indirizzi IP elastici in Amazon EC2

Il seguente esempio di codice:

1. Visualizza informazioni su qualsiasi indirizzo associato a un'istanza Amazon Elastic Compute Cloud (Amazon EC2).
2. Crea un indirizzo IP elastico in Amazon Virtual Private Cloud (Amazon VPC).
3. Associa l'indirizzo all'istanza.
4. Visualizza nuovamente le informazioni sugli indirizzi associati all'istanza. Questa volta, dovrebbe essere visualizzata la nuova associazione di indirizzi.
5. Rilascia l'indirizzo.
6. Visualizza nuovamente le informazioni sugli indirizzi associati all'istanza. Questa volta, l'indirizzo rilasciato non dovrebbe essere visualizzato.

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

# This code example does the following:
# 1. Displays information about any addresses associated with an
#    Amazon Elastic Compute Cloud (Amazon EC2) instance.
# 2. Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
# 3. Associates the address with the instance.
# 4. Displays information again about addresses associated with the instance.
#    This time, the new address association should display.
# 5. Releases the address.
# 6. Displays information again about addresses associated with the instance.
#    This time, the released address should not display.

require 'aws-sdk-ec2'

# Checks whether the specified Amazon Elastic Compute Cloud
# (Amazon EC2) instance exists.
#
# 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 exists; otherwise, false.
# @example
#   exit 1 unless instance_exists?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_exists?(ec2_client, instance_id)
  ec2_client.describe_instances(instance_ids: [instance_id])
  return true
rescue StandardError
  return false
end

# 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-east-1'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return 'Error'
end

# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id,
  )
  return response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return 'Error'
end

# Gets information about addresses associated with 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.
# @example
#   describe_addresses_for_instance(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def describe_addresses_for_instance(ec2_client, instance_id)
  response = ec2_client.describe_addresses(
    filters: [
      {
        name: 'instance-id',
        values: [instance_id]
      }
    ]
  )
  addresses = response.addresses
  if addresses.count.zero?
    puts 'No addresses.'
  else
    addresses.each do |address|
      puts '-' * 20
      puts "Public IP:  #{address.public_ip}"
      puts "Private IP: #{address.private_ip_address}"
    end
  end
rescue StandardError => e
  puts "Error getting address information for instance: #{e.message}"
end

# 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-east-1'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  return "Error releasing Elastic IP address: #{e.message}"
  return false
end

# Full example call:
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-elastic-ips.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-elastic-ips.rb ' \
      'i-033c48ef067af3dEX us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-033c48ef067af3dEX'
    region = 'us-east-1'
  # 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)

  unless instance_exists?(ec2_client, instance_id)
    puts "Cannot find instance with ID '#{instance_id}'. Stopping program."
    exit 1
  end
```

```
  puts "Addresses for instance with ID '#{instance_id}' before allocating " \
    'Elastic IP address:'
  describe_addresses_for_instance(ec2_client, instance_id)

  puts 'Allocating Elastic IP address...'
  allocation_id = allocate_elastic_ip_address(ec2_client)
  if allocation_id.start_with?('Error')
    puts 'Stopping program.'
    exit 1
  else
    puts "Elastic IP address created with allocation ID '#{allocation_id}'."
  end

  puts 'Associating Elastic IP address with instance...'
  association_id = associate_elastic_ip_address_with_instance(
    ec2_client,
    allocation_id,
    instance_id
  )
  if association_id.start_with?('Error')
    puts 'Stopping program. You must associate the Elastic IP address yourself.'
    exit 1
  else
    puts 'Elastic IP address associated with instance with association ID ' \
      "'#{association_id}'."
  end

  puts 'Addresses for instance after allocating Elastic IP address:'
  describe_addresses_for_instance(ec2_client, instance_id)

  puts 'Releasing the Elastic IP address from the instance...'
  if elastic_ip_address_released?(ec2_client, allocation_id) == false
    puts 'Stopping program. You must release the Elastic IP address yourself.'
    exit 1
  else
    puts 'Address released.'
  end

  puts 'Addresses for instance after releasing Elastic IP address:'
  describe_addresses_for_instance(ec2_client, instance_id)
end
```

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

Creazione di un gruppo di sicurezza Amazon EC2

Il seguente esempio di codice crea un gruppo di sicurezza Amazon EC2 e quindi aggiunge una regola in uscita a quel gruppo di sicurezza.

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

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group and
# then adds an outbound rule to that security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon EC2 resource object.
# @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.
# @param protocol [String] The network protocol for the outbound rule.
# @param from_port [String] The originating port for the outbound rule.
# @param to_port [String] The destination port for the outbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the outbound rule.
# @return [Boolean] true if the security group was created and the outbound
#   rule was added; otherwise, false.
# @example
#   exit 1 unless security_group_created_with_egress?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX',
#     'tcp',
#     '22',
#     '22',
#     '0.0.0.0/0'
#   )
def security_group_created_with_egress?(
  ec2_resource,
```

```
  group_name,
  description,
  vpc_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  security_group = ec2_resource.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.id}' in VPC with ID '#{vpc_id}'."
  security_group.authorize_egress(
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Granted egress to security group '#{group_name}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error creating security group or granting egress: #{e.message}"
  return false
end

# Full example call:
def run_me
  group_name = ''
  description = ''
  vpc_id = ''
  ip_protocol = ''
```

```
from_port = ''
to_port = ''
cidr_ip_range = ''
region = ''
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage: ruby ec2-ruby-example-create-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL FROM_PORT TO_PORT ' \
    'CIDR_IP_RANGE REGION'
  puts 'Example: ruby ec2-ruby-example-create-security-group.rb ' \
    'my-security-group \'This is my security group.\' vpc-6713dfEX ' \
    'tcp 22 22 \'0.0.0.0/0\' us-east-1'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  group_name = 'my-security-group'
  description = 'This is my security group.'
  vpc_id = 'vpc-6713dfEX'
  ip_protocol = 'tcp'
  from_port = '22'
  to_port = '22'
  cidr_ip_range = '0.0.0.0/0'
  region = 'us-east-1'
# Otherwise, use the values as specified at the command prompt.
else
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol = ARGV[3]
  from_port = ARGV[4]
  to_port = ARGV[5]
  cidr_ip_range = ARGV[6]
  region = ARGV[7]
end

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

if security_group_created_with_egress?(
  ec2_resource,
  group_name,
  description,
  vpc_id,
  ip_protocol,
  from_port,
```

```
    to_port,
    cidr_ip_range
  )
    puts 'Security group created and egress granted.'
  else
    puts 'Security group not created or egress not granted.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Lavorare con i gruppi di sicurezza di Amazon EC2

L'esempio seguente:

1. Crea un gruppo di sicurezza Amazon EC2.
2. Aggiunge regole in entrata al gruppo di sicurezza.
3. Visualizza informazioni sui gruppi di sicurezza disponibili.
4. Elimina il gruppo di sicurezza.

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

# 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-east-1'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return 'Error'
end

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

```
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client,
  security_group_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

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

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

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

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

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

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

  print "\n"
end

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

```
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-east-1'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      puts '-' * (sg.group_name.length + 13)
      puts "Name:        #{sg.group_name}"
      puts "Description: #{sg.description}"
      puts "Group ID:    #{sg.group_id}"
      puts "Owner ID:    #{sg.owner_id}"
      puts "VPC ID:      #{sg.vpc_id}"

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

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

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

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
```

```
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'sg-030a858e078f1b9EX'
#   )
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Full example call:
def run_me
  group_name = ''
  description = ''
  vpc_id = ''
  ip_protocol_http = ''
  from_port_http = ''
  to_port_http = ''
  cidr_ip_range_http = ''
  ip_protocol_ssh = ''
  from_port_ssh = ''
  to_port_ssh = ''
  cidr_ip_range_ssh = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
      'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
      'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
      'CIDR_IP_RANGE_2 REGION'
```

```
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    'my-security-group \'This is my security group.\' vpc-6713dfEX ' \
    'tcp 80 80 \'0.0.0.0/0\' tcp 22 22 \'0.0.0.0/0\' us-east-1'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  group_name = 'my-security-group'
  description = 'This is my security group.'
  vpc_id = 'vpc-6713dfEX'
  ip_protocol_http = 'tcp'
  from_port_http = '80'
  to_port_http = '80'
  cidr_ip_range_http = '0.0.0.0/0'
  ip_protocol_ssh = 'tcp'
  from_port_ssh = '22'
  to_port_ssh = '22'
  cidr_ip_range_ssh = '0.0.0.0/0'
  region = 'us-east-1'
# Otherwise, use the values as specified at the command prompt.
else
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

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

puts 'Attempting to create security group...'
security_group_id = create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
```

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

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

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

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

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

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

Utilizzo delle coppie di chiavi in Amazon EC2

Il seguente esempio di codice:

1. Crea una coppia di key pair in Amazon EC2.
2. Visualizza informazioni sulle coppie di chiavi disponibili.
3. Elimina la key pair.

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

# 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'

# Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2) and
# saves the resulting RSA private key file locally in the calling
# user's home directory.
#
# @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-east-1'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + '.pem')
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
```

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

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-east-1'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-key-pair'
```

```
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return false
end

# Full example call:
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-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-east-1'
  # 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__
```

Ottenere informazioni su tutte le istanze Amazon EC2

Il seguente esempio di codice elenca gli ID e gli stati correnti delle istanze Amazon EC2 disponibili.

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

require 'aws-sdk-ec2'

# Lists the IDs and current states of available
# Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-east-1'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
```

```
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

#Full example call:
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'
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.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
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

Ottenere informazioni su tutte le istanze Amazon EC2 con un valore di tag specifico

Il seguente esempio di codice elenca gli ID e gli stati correnti delle istanze Amazon EC2 disponibili che corrispondono alla chiave e al valore del tag specificati.

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

require 'aws-sdk-ec2'
```

```
# Lists the IDs, current states, and tag keys/values of matching
# available Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @param tag_key [String] The key portion of the tag to search on.
# @param tag_value [String] The value portion of the tag to search on.
# @example
#   list_instance_ids_states_by_tag(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'my-key',
#     'my-value'
#   )
def list_instance_ids_states_by_tag(ec2_resource, tag_key, tag_value)
  response = ec2_resource.instances(
    filters: [
      {
        name: "tag:#{tag_key}",
        values: [tag_value]
      }
    ]
  )
  if response.count.zero?
    puts 'No matching instances found.'
  else
    puts 'Matching instances -- ID, state, tag key/value:'
    response.each do |instance|
      print "#{instance.id}, #{instance.state.name}"
      instance.tags.each do |tag|
        print ", #{tag.key}/#{tag.value}"
      end
      print "\n"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

#Full example call:
def run_me
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
```

```
    puts 'Usage:   ruby ec2-ruby-example-get-instance-info-by-tag.rb ' \
      'TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-get-instance-info-by-tag.rb ' \
      'my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    tag_key = ARGV[0]
    tag_value = ARGV[1]
    region = ARGV[2]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states_by_tag(ec2_resource, tag_key, tag_value)
end

run_me if $PROGRAM_NAME == __FILE__
```

Ottenere informazioni su un'istanza Amazon EC2 specifica

L'esempio seguente elenca lo stato dell'istanza Amazon EC2 specificata.

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

require 'aws-sdk-ec2'

# Lists the state of an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @example
#   list_instance_state(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'i-123abc'
```

```
#   )
def list_instance_state(ec2_client, instance_id)
  response = ec2_client.describe_instances(
    instance_ids: [instance_id]
  )
  if response.count.zero?
    puts 'No matching instance found.'
  else
    instance = response.reservations[0].instances[0]
    puts "The instance with ID '#{instance_id}' is '#{instance.state.name}'."
  end
rescue StandardError => e
  puts "Error getting information about instance: #{e.message}"
end

# Full example call:
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-list-state-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-list-state-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # 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)
  list_instance_state(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creazione di un'istanza Amazon EC2

L'esempio seguente crea e contrassegna un'istanza Amazon EC2.

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

require 'aws-sdk-ec2'
require 'base64'

# Creates and tags an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An EC2 key pair.
# - If you want to run any commands on the instance after it starts, a
#   file containing those commands.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @param image_id [String] The ID of the target Amazon Machine Image (AMI).
# @param key_pair_name [String] The name of the existing EC2 key pair.
# @param tag_key [String] The key portion of the tag for the instance.
# @param tag_value [String] The value portion of the tag for the instance.
# @param instance_type [String] The ID of the type of instance to create.
#   If not specified, the default value is 't2.micro'.
# @param user_data_file [String] The path to the file containing any commands
#   to run on the instance after it starts. If not specified, the default
#   value is an empty string.
# @return [Boolean] true if the instance was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless instance_created?(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'ami-0947d2ba12EXAMPLE',
#     'my-key-pair',
#     'my-key',
#     'my-value',
#     't2.micro',
#     'my-user-data.txt'
#   )
def instance_created?(
  ec2_resource,
  image_id,
  key_pair_name,
```

```
  tag_key,
  tag_value,
  instance_type = 't2.micro',
  user_data_file = ''
)
  encoded_script = ''

  unless user_data_file == ''
    script = File.read(user_data_file)
    encoded_script = Base64.encode64(script)
  end

  instance = ec2_resource.create_instances(
    image_id: image_id,
    min_count: 1,
    max_count: 1,
    key_name: key_pair_name,
    instance_type: instance_type,
    user_data: encoded_script
  )

  puts 'Creating instance...'

  # Check whether the new instance is in the "running" state.
  polls = 0
  loop do
    polls += 1
    response = ec2_resource.client.describe_instances(
      instance_ids: [
        instance.first.id
      ]
    )
    # Stop polling after 10 minutes (40 polls * 15 seconds per poll) if not running.
    break if response.reservations[0].instances[0].state.name == 'running' || polls >
 40

    sleep(15)
  end

  puts "Instance created with ID '#{instance.first.id}'."

  instance.batch_create_tags(
    tags: [
      {
```

```
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Instance tagged.'

  return true
rescue StandardError => e
  puts "Error creating or tagging instance: #{e.message}"
  return false
end

# Full example call:
def run_me
  image_id = ''
  key_pair_name = ''
  tag_key = ''
  tag_value = ''
  instance_type = ''
  region = ''
  user_data_file = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-instance.rb ' \
      'IMAGE_ID KEY_PAIR_NAME TAG_KEY TAG_VALUE INSTANCE_TYPE ' \
      'REGION [USER_DATA_FILE]'
    puts 'Example: ruby ec2-ruby-example-create-instance.rb ' \
      'ami-0947d2ba12EXAMPLE my-key-pair my-key my-value t2.micro ' \
      'us-east-1 my-user-data.txt'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    image_id = 'ami-0947d2ba12EXAMPLE'
    key_pair_name = 'my-key-pair'
    tag_key = 'my-key'
    tag_value = 'my-value'
    instance_type = 't2.micro'
    region = 'us-east-1'
    user_data_file = 'my-user-data.txt'
  # Otherwise, use the values as specified at the command prompt.
  else
    image_id = ARGV[0]
    key_pair_name = ARGV[1]
```

```
    tag_key = ARGV[2]
    tag_value = ARGV[3]
    instance_type = ARGV[4]
    region = ARGV[5]
    user_data_file = ARGV[6] if ARGV.count == 7 # If user data file specified.
  end

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

  if instance_created?(
    ec2_resource,
    image_id,
    key_pair_name,
    tag_key,
    tag_value,
    instance_type,
    user_data_file
  )
    puts 'Created and tagged instance.'
  else
    puts 'Could not create or tag instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Arresto di un'istanza Amazon EC2

L'esempio seguente tenta di arrestare l'istanza Amazon EC2 specificata.

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

require 'aws-sdk-ec2'

# Attempts to stop 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 stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end

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

# Full example call:
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 '
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
```

```
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

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

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_stopped?(ec2_client, instance_id)
    puts 'Could not stop instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Avvio di un'istanza Amazon EC2

L'esempio seguente tenta di avviare l'istanza Amazon EC2 specificata.

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

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-east-1'),
```

```
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return false
end

# Full example call:
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 '
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
```

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

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

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_started?(ec2_client, instance_id)
    puts 'Could not start instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Riavvio di un'istanza Amazon EC2

L'esempio seguente tenta di riavviare l'istanza Amazon EC2 specificata.

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

require 'aws-sdk-ec2'

# Reboots an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @example
#   request_instance_reboot(
#     Aws::EC2::Resource.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def request_instance_reboot(ec2_client, instance_id)
  response = ec2_client.describe_instances(instance_ids: [instance_id])
  if response.count.zero?
```

```
    puts 'Error requesting reboot: no matching instance found.'
  else
    instance = response.reservations[0].instances[0]
    if instance.state.name == 'terminated'
      puts 'Error requesting reboot: the instance is already terminated.'
    else
      ec2_client.reboot_instances(instance_ids: [instance_id])
      puts 'Reboot request sent.'
    end
  end
rescue StandardError => e
  puts "Error requesting reboot: #{e.message}"
end

# Full example call:
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-reboot-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-reboot-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # 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)
  request_instance_reboot(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Gestione delle istanze Amazon EC2

Il seguente esempio di codice:

1. Interrompe un'istanza Amazon EC2.
2. Riavvia l'istanza.
3. Riavvia l'istanza.
4. Abilita il monitoraggio dettagliato dell'istanza.
5. Visualizza informazioni sulle istanze disponibili.

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

# This code example does the following:
# 1. Stops an Amazon Elastic Compute Cloud (Amazon EC2) instance.
# 2. Restarts the instance.
# 3. Reboots the instance.
# 4. Enables detailed monitoring for the instance.
# 5. Displays information about available instances.

require 'aws-sdk-ec2'

# Waits for an Amazon Elastic Compute Cloud (Amazon EC2) instance
# to reach the specified state.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_state [Symbol] The desired instance state.
# @param instance_id [String] The ID of the instance.
# @example
#   wait_for_instance(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     :instance_stopped,
#     'i-033c48ef067af3dEX'
#   )
def wait_for_instance(ec2_client, instance_state, instance_id)
  ec2_client.wait_until(instance_state, instance_ids: [instance_id])
  puts "Success: #{instance_state}."
```

```
rescue Aws::Waiters::Errors::WaiterFailed => e
  puts "Failed: #{e.message}"
end

# Attempts to stop 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 stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_stopped?(ec2_client, instance_id)
  ec2_client.stop_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_stopped, instance_id)
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Attempts to restart 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 restarted; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_restarted?(ec2_client, instance_id)
  ec2_client.start_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_running, instance_id)
```

```
  return true
rescue StandardError => e
  puts "Error restarting instance: #{e.message}"
  return false
end

# Attempts to reboot 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 rebooted; otherwise, false.
# @example
#   exit 1 unless instance_rebooted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_rebooted?(ec2_client, instance_id)
  ec2_client.reboot_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_status_ok, instance_id)
  return true
rescue StandardError => e
  puts "Error rebooting instance: #{e.message}"
  return false
end

# Attempts to enabled detailed monitoring for 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 detailed monitoring was enabled; otherwise, false.
# @example
#   exit 1 unless instance_detailed_monitoring_enabled?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
```

```
def instance_detailed_monitoring_enabled?(ec2_client, instance_id)
  result = ec2_client.monitor_instances(instance_ids: [instance_id])
  puts "Detailed monitoring state: #{result.instance_monitorings[0].monitoring.state}"
  return true
rescue Aws::EC2::Errors::InvalidState
  puts "The instance is not in a monitorable state. Skipping this step."
  return false
rescue StandardError => e
  puts "Error enabling detailed monitoring: #{e.message}"
  return false
end

# Displays information about available
# Amazon Elastic Compute Cloud (Amazon EC2) instances.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_instances_information(Aws::EC2::Client.new(region: 'us-east-1'))
def list_instances_information(ec2_client)
  result = ec2_client.describe_instances
  result.reservations.each do |reservation|
    if reservation.instances.count.positive?
      reservation.instances.each do |instance|
        puts '-' * 12
        puts "Instance ID:               #{instance.instance_id}"
        puts "State:                     #{instance.state.name}"
        puts "Image ID:                  #{instance.image_id}"
        puts "Instance type:             #{instance.instance_type}"
        puts "Architecture:              #{instance.architecture}"
        puts "IAM instance profile ARN:  #{instance.iam_instance_profile.arn}"
        puts "Key name:                  #{instance.key_name}"
        puts "Launch time:               #{instance.launch_time}"
        puts "Detailed monitoring state: #{instance.monitoring.state}"
        puts "Public IP address:         #{instance.public_ip_address}"
        puts "Public DNS name:           #{instance.public_dns_name}"
        puts "VPC ID:                    #{instance.vpc_id}"
        puts "Subnet ID:                 #{instance.subnet_id}"
        if instance.tags.count.positive?
          puts 'Tags:'
          instance.tags.each do |tag|
            puts "                           #{tag.key}/#{tag.value}"
          end
        end
      end
```

```
    end
  end
end

# Full example call:
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-manage-instances.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-manage-instances.rb ' \
      'i-033c48ef067af3dEX us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-033c48ef067af3dEX'
    region = 'us-east-1'
  # 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 the instance. ' \
    'This might take a few minutes...'
  unless instance_stopped?(ec2_client, instance_id)
    puts 'Cannot stop the instance. Skipping this step.'
  end

  puts "\nAttempting to restart the instance. " \
    'This might take a few minutes...'
  unless instance_restarted?(ec2_client, instance_id)
    puts 'Cannot restart the instance. Skipping this step.'
  end

  puts "\nAttempting to reboot the instance. " \
    'This might take a few minutes...'
  unless instance_rebooted?(ec2_client, instance_id)
    puts 'Cannot reboot the instance. Skipping this step.'
  end
```

```
  puts "\nAttempting to enable detailed monitoring for the instance..."
  unless instance_detailed_monitoring_enabled?(ec2_client, instance_id)
    puts 'Cannot enable detailed monitoring for the instance. ' \
      'Skipping this step.'
  end

  puts "\nInformation about available instances:"
  list_instances_information(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Terminazione di un'istanza Amazon EC2

L'esempio seguente tenta di terminare l'istanza Amazon EC2 specificata.

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

require 'aws-sdk-ec2'

# Attempts to terminate 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 terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
```

```
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return false
end

# Full example call:
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 '
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

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

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  unless instance_terminated?(ec2_client, instance_id)
    puts 'Could not terminate instance.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Ottenere informazioni su regioni e zone di disponibilità per Amazon EC2

L'esempio seguente:

1. Visualizza un elenco Regioni AWS di Amazon EC2 disponibili per te.
2. Visualizza un elenco di zone di disponibilità di Amazon EC2 disponibili a seconda Regione AWS del client Amazon EC2.

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

require 'aws-sdk-ec2'

# Displays a list of AWS Regions for Amazon Elastic Compute Cloud (Amazon EC2)
# that are available to you.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-east-1'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name.to_s
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint.to_s
    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-east-1'))
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

# Full example call:
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'
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.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

  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__
```

AWS Elastic BeanstalkEsempi di utilizzo dell'AWSSDK for Ruby

AWS Elastic Beanstalkti consente di distribuire e gestire rapidamente le applicazioni nel AWS cloud senza preoccuparti dell'infrastruttura che esegue tali applicazioni. È possibile utilizzare i seguenti esempi per accedere a Elastic Beanstalk utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Elastic Beanstalk, consulta la documentazione. AWS Elastic Beanstalk

Argomenti

- Ottenere informazioni su tutte le applicazioni in AWS Elastic Beanstalk
- Ottenere informazioni su un'applicazione specifica in AWS Elastic Beanstalk
- Aggiornamento di un'applicazione Ruby on Rails per AWS Elastic Beanstalk

Ottenere informazioni su tutte le applicazioni in AWS Elastic Beanstalk

L'esempio seguente elenca i nomi, le descrizioni e gli URL di tutte le applicazioni Elastic Beanstalk nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

eb = Aws::ElasticBeanstalk::Client.new(region: 'us-west-2')

eb.describe_applications.applications.each do |a|
  puts "Name:         #{a.application_name}"
  puts "Description:  #{a.description}"

  eb.describe_environments({application_name: a.application_name}).environments.each do
 |env|
    puts "  Environment:  #{env.environment_name}"
    puts "    URL:        #{env.cname}"
    puts "    Health:     #{env.health}"
  end
end
```

Ottenere informazioni su un'applicazione specifica in AWS Elastic Beanstalk

L'esempio seguente elenca il nome, la descrizione e l'URL dell'`MyRailsApp`applicazione nella `us-west-2` regione.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
```

```
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

eb = Aws::ElasticBeanstalk::Client.new(region: 'us-west-2')

app = eb.describe_applications({application_names: [args[0]]})

if app.exists?
  puts "Name:         #{app.application_name}"
  puts "Description:  #{app.description}"

  envs = eb.describe_environments({application_name: app.application_name})
  puts "URL:          #{envs.environments[0].cname}"
end
```

Aggiornamento di un'applicazione Ruby on Rails per AWS Elastic Beanstalk

L'esempio seguente aggiorna l'applicazione Ruby on Rails nella regione. `MyRailsApp` `us-west-2`

> **Note**
>
> Devi essere nella directory principale della tua app Rails per eseguire correttamente lo script.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.
```

```
require 'aws-sdk-elasticbeanstalk'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

eb = Aws::ElasticBeanstalk::Client.new
s3 = Aws::S3::Client.new

app_name = 'MyRailsApp'

# Get S3 bucket containing app
app_versions = eb.describe_application_versions({ application_name: app_name })
av = app_versions.application_versions[0]
bucket = av.source_bundle.s3_bucket
s3_key = av.source_bundle.s3_key

# Get info on environment
envs = eb.describe_environments({ application_name: app_name })
env = envs.environments[0]
env_name = env.environment_name

# Create new storage location
resp = eb.create_storage_location()

puts "Created storage location in bucket #{resp.s3_bucket}"

s3.list_objects({
  prefix: s3_key,
  bucket: bucket
})

# Create ZIP file
zip_file_basename = SecureRandom.urlsafe_base64.to_s
zip_file_name = zip_file_basename + '.zip'

# Call out to OS to produce ZIP file
cmd = "git archive --format=zip -o #{zip_file_name} HEAD"
%x[ #{cmd} ]

# Get ZIP file contents
zip_contents = File.read(zip_file_name)

key = app_name + "\\" + zip_file_name

s3.put_object({
```

```
  body: zip_contents,
  bucket: bucket,
  key: key
})

date = Time.new
today = date.day.to_s + "/" + date.month.to_s + "/" + date.year.to_s

eb.create_application_version({
  process: false,
  application_name: app_name,
  version_label: zip_file_basename,
  source_bundle: {
    s3_bucket: bucket,
    s3_key: key
  },
  description: "Updated #{today}"
})

eb.update_environment({
  environment_name: env_name,
  version_label: zip_file_basename
})
```

AWS Identity and Access Management(IAM) Esempi di utilizzo dell'AWSSDK for Ruby

AWS Identity and Access Management(IAM) è un servizio web per il controllo sicuro degli accessi a. Servizi AWS Puoi usare i seguenti esempi per accedere a IAM utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su IAM, consulta la documentazione IAM.

Argomenti

- Ottenere informazioni sugli utenti IAM
- Elenco degli utenti IAM che sono amministratori
- Aggiungere un nuovo utente IAM
- Crea chiavi di accesso utente per un utente IAM
- Aggiungere una policy gestita a un utente IAM
- Creazione di un ruolo IAM
- Gestione degli utenti IAM

- Lavorare con le policy IAM;
- Gestione delle chiavi di accesso IAM
- Utilizzo dei certificati server IAM;
- Gestire gli alias per l'account IAM

Ottenere informazioni sugli utenti IAM

L'esempio seguente elenca i gruppi, le politiche e gli ID delle chiavi di accesso degli utenti IAM nella `us-west-2` regione. Se ci sono più di 100 utenti, `iam.list_users.IsTruncated` è vero e `iam.list_users.Marker` contiene un valore che è possibile utilizzare per ottenere informazioni su altri utenti. Per ulteriori informazioni, consulta l'argomento Aws: :IAM: :Client.list_users.

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

require 'aws-sdk-iam'

# Displays information about available users in
# AWS Identity and Access Management (IAM) including users'
# names, associated group names, inline embedded user policy names,
# and access key IDs.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   get_user_details(Aws::IAM::Client.new)
def get_user_details(iam_client)
  users_response = iam_client.list_users

  if users_response.key?('users') && users_response.users.count.positive?

    # Are there more users available than can be displayed?
    if users_response.key?('is_truncated') && users_response.is_truncated
      puts '(Note: not all users are displayed here, ' \
        "only the first #{users_response.users.count}.)"
    else
      puts "Found #{users_response.users.count} user(s):"
    end

    users_response.users.each do |user|
      name = user.user_name
      puts '-' * 30
```

```
      puts "User name: #{name}"

      puts "Groups:"
      groups_response = iam_client.list_groups_for_user(user_name: name)
      if groups_response.key?('groups') &&
        groups_response.groups.count.positive?

        groups_response.groups.each do |group|
          puts "  #{group.group_name}"
        end
      else
        puts '  None'
      end

      puts 'Inline embedded user policies:'
      policies_response = iam_client.list_user_policies(user_name: name)
      if policies_response.key?('policy_names') &&
        policies_response.policy_names.count.positive?

        policies_response.policy_names.each do |policy_name|
          puts "  #{policy_name}"
        end
      else
        puts '  None'
      end

      puts 'Access keys:'
      access_keys_response = iam_client.list_access_keys(user_name: name)

      if access_keys_response.key?('access_key_metadata') &&
        access_keys_response.access_key_metadata.count.positive?

        access_keys_response.access_key_metadata.each do |access_key|
          puts "  #{access_key.access_key_id}"
        end
      else
        puts '  None'
      end
    end
  else
    puts 'No users found.'
  end
rescue StandardError => e
  puts "Error getting user details: #{e.message}"
```

```
end

# Full example call:
def run_me
  iam_client = Aws::IAM::Client.new
  puts 'Attempting to get details for available users...'
  get_user_details(iam_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

Elenco degli utenti IAM che sono amministratori

L'esempio seguente utilizza il metodo get_account_authorization_details, per ottenere l'elenco degli utenti per l'account corrente.

Scegliete di salvare il codice localmente. Copy

Crea il file get_admins.rb.

Aggiungi la gem IAM richiesta e la gem os e usa quest'ultima per utilizzare il certificato in bundle se utilizzi Microsoft Windows.

> **Note**
>
> La versione 2 dell'AWSSDK for Ruby non conteneva gemme specifiche del servizio.

```
require 'aws-sdk-iam'  # v2: require 'aws-sdk'
require 'os'

if OS.windows?
 Aws.use_bundled_cert!
end
```

Crea un metodo per determinare se l'utente dispone di una politica con privilegi di amministratore.

```
def user_has_admin_policy(user, admin_access)
  policies = user.user_policy_list

  policies.each do |p|
```

```
    if p.policy_name == admin_access
      return true
    end
  end

  false
end
```

Crea un metodo per determinare se l'utente dispone di una politica allegata con privilegi di amministratore.

```
def user_has_attached_policy(user, admin_access)
  attached_policies = user.attached_managed_policies

  attached_policies.each do |p|
    if p.policy_name == admin_access
      return true
    end
  end

  false
end
```

Crea un metodo per determinare se un gruppo a cui appartiene l'utente dispone di una politica con privilegi di amministratore.

```
```

Crea un metodo per determinare se un gruppo a cui appartiene l'utente ha una politica allegata con privilegi di amministratore.

```
def group_has_admin_policy(client, group, admin_access)
 resp = client.list_group_policies(
   group_name: group.group_name
 )

 resp.policy_names.each do |name|
   if name == admin_access
     return true
   end
 end
```

```
 false
end
```

Crea un metodo per determinare se un gruppo a cui appartiene l'utente dispone dei privilegi di amministratore.

```
def user_has_admin_from_group(client, user, admin_access)
 resp = client.list_groups_for_user(
   user_name: user.user_name
 )

 resp.groups.each do |group|
   has_admin_policy = group_has_admin_policy(client, group, admin_access)
   if has_admin_policy
     return true
   end

   has_attached_policy = group_has_attached_policy(client, group, admin_access)
   if has_attached_policy
     return true
   end
 end

 false
end
```

Crea un metodo per determinare se l'utente dispone dei privilegi di amministratore.

```
def is_user_admin(client, user, admin_access)
 has_admin_policy = user_has_admin_policy(user, admin_access)
 if has_admin_policy
   return true
 end

 has_attached_admin_policy = user_has_attached_policy(user, admin_access)
 if has_attached_admin_policy
   return true
 end

 has_admin_from_group = user_has_admin_from_group(client, user, admin_access)
 if has_admin_from_group
   return true
 end
```

```
 false
end
```

Crea un metodo per scorrere un elenco di utenti e indicare quanti di questi utenti dispongono dei privilegi di amministratore.

```
<code>
```

La routine principale inizia qui. Crea un client IAM e delle variabili per memorizzare il numero di utenti, il numero di utenti con privilegi di amministratore e la stringa che identifica una politica che fornisce i privilegi di amministratore.

```
def get_admin_count(client, users, admin_access)
 num_admins = 0

 users.each do |user|
   is_admin = is_user_admin(client, user, admin_access)
   if is_admin
     puts user.user_name
     num_admins += 1
   end
 end

 num_admins
end
```

Chiama `get_account_authorization_details` per ottenere i dettagli dell'account e ottenere gli utenti per l'account. `user_detail_list` Tieni traccia del numero di utenti che riceviamo, chiama `get_admin_count` per conoscere il numero di utenti con privilegi di amministratore e tieni traccia del numero di tali utenti.

```
details = client.get_account_authorization_details(
 filter: ['User']
)

users = details.user_detail_list
num_users += users.count
more_admins = get_admin_count(client, users, access_admin)
num_admins += more_admins
```

Se la prima chiamata a non `get_account_authorization_details` ha ricevuto tutti i dettagli, richiamala e ripeti la procedura per determinare quanti hanno i privilegi di amministratore.

```
<code>
```

Infine, mostra quanti utenti hanno i privilegi di amministratore.

```

more_users = details.is_truncated

mentre more_users
    details = client.get_account_authorization_details (
        filter: ['Utente'], marker: details.marker
    )
    utenti = details.user_detail_list
    num_users += users.count more_admins = get_admin_count (cliente, utenti, access_admin)
    num_admins += more_admins
    more_users = details.is_troncated

fine
```

Vedi l'esempio completo su GitHub.

Aggiungere un nuovo utente IAM

L'esempio seguente crea l'utente IAM `my_groovy_user` nella `us-west-2` regione con la password `REPLACE_ME` e visualizza l'ID dell'account dell'utente. Se esiste già un utente con quel nome, visualizza un messaggio e non crea un nuovo utente.

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

require 'aws-sdk-iam'

# Creates a user in AWS Identity and Access Management (IAM).
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
```

```
# @param user_name [String] The name of the user.
# @param initial_password [String] The initial password for the user.
# @return [String] The ID of the user if the user was created, otherwise;
#   the string 'Error'.
# @example
#   puts create_user(Aws::IAM::Client.new, 'my-user', 'my-!p@55w0rd!')
def create_user(iam_client, 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(
    password: initial_password,
    password_reset_required: true,
    user_name: user_name
  )
  return response.user.user_id
rescue Aws::IAM::Errors::EntityAlreadyExists
  puts "Error creating user '#{user_name}': user already exists."
  return 'Error'
rescue StandardError => e
  puts "Error creating user '#{user_name}': #{e.message}"
  return 'Error'
end

# Full example call:
def run_me
  user_name = 'my-user'
  initial_password = 'my-!p@55w0rd!'
  iam_client = Aws::IAM::Client.new

  puts "Attempting to create user '#{user_name}'..."
  user_id = create_user(iam_client, user_name, initial_password)

  if user_id == 'Error'
    puts 'User not created.'
  else
    puts "User '#{user_name}' created with ID '#{user_id}' and initial " \
      "sign-in password '#{initial_password}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Crea chiavi di accesso utente per un utente IAM

L'esempio seguente crea una chiave di accesso e una chiave segreta per l'utente IAM `my_groovy_user` nella `us-west-2` regione.

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

require 'aws-sdk-iam'

# Creates an access key for a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   create_access_key(Aws::IAM::Client.new, 'my-user')
def create_access_key(iam, user_name)
  response = iam.create_access_key(user_name: user_name)
  access_key = response.access_key
  puts 'Access key created:'
  puts "  Access key ID: #{access_key.access_key_id}"
  puts "  Secret access key: #{access_key.secret_access_key}"
  puts 'Keep a record of this information in a secure location. ' \
    'This will be the only time you will be able to view the ' \
    'secret access key.'
rescue Aws::IAM::Errors::LimitExceeded
  puts 'Error creating access key: limit exceeded. Cannot create any more. ' \
    'To create more, delete an existing access key, and then try again.'
rescue StandardError => e
  puts "Error creating access key: #{e.message}"
end

# Full example call:
def run_me
  iam = Aws::IAM::Client.new
  user_name = 'my-user'

  puts 'Attempting to create an access key...'
  create_access_key(iam, user_name)
end
```

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

Aggiungere una policy gestita a un utente IAM

L'esempio seguente aggiunge la policy gestita `AmazonS3FullAccess` all'utente IAM `my_groovy_user` nella `us-west-2` regione.

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

require 'aws-sdk-iam'

# Attaches a policy to a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @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 the policy was attached; otherwise, false.
# @example
#   exit 1 unless alias_created?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'arn:aws:iam::aws:policy/AmazonS3FullAccess'
#   )
def policy_attached_to_user?(iam_client, user_name, policy_arn)
  iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  return true
rescue StandardError => e
  puts "Error attaching policy to user: #{e.message}"
  return false
end

# Full example call:
def run_me
  user_name = 'my-user'
  arn_prefix = 'arn:aws:iam::aws:policy/'
```

```
  policy_arn = arn_prefix + 'AmazonS3FullAccess'
  iam_client = Aws::IAM::Client.new

  puts "Attempting to attach policy with ARN '#{policy_arn}' to " \
    "user '#{user_name}'..."

  if policy_attached_to_user?(iam_client, user_name, policy_arn)
    puts 'Policy attached.'
  else
    puts 'Policy not attached.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Creazione di un ruolo IAM

L'esempio seguente crea il ruolo `my_groovy_role` in modo che Amazon EC2 possa accedere ad Amazon S3 e Amazon DynamoDB nella regione. `us-west-2`

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

require 'aws-sdk-iam'

# Creates a role in AWS Access and Identity Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] A name for the role.
# @param assume_role_policy_document [String]
# @param policy_arns [Array] An array of type String representing
#   Amazon Resource Names (ARNs) corresponding to available
#   IAM managed policies.
# @return [String] The ARN of the new role; otherwise, the string 'Error'.
# @example
#   puts create_role(
#     Aws::IAM::Client.new,
#     'my-ec2-s3-dynamodb-full-access-role',
#     {
#       Version: '2012-10-17',
#       Statement: [
#         {
#           Effect: 'Allow',
```

```
#           Principal: {
#             Service: 'ec2.amazonaws.com'
#           },
#           Action: 'sts:AssumeRole'
#         }
#       ]
#     },
#     [
#       'arn:aws:iam::aws:policy/AmazonS3FullAccess',
#       'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess'
#     ]
#   )
def create_role(
  iam_client,
  role_name,
  assume_role_policy_document,
  policy_arns
)
  iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: assume_role_policy_document.to_json
  )
  policy_arns.each do |policy_arn|
    iam_client.attach_role_policy(
      policy_arn: policy_arn,
      role_name: role_name,
    )
  end
  return iam_client.get_role(role_name: role_name).role.arn
rescue StandardError => e
  puts "Error creating role: #{e.message}"
  return 'Error'
end

# Full example call:
def run_me
  role_name = 'my-ec2-s3-dynamodb-full-access-role'

  # Allow the role to trust Amazon Elastic Compute Cloud (Amazon EC2)
  # within the AWS account.
  assume_role_policy_document = {
    Version: '2012-10-17',
    Statement: [
      {
```

```
        Effect: 'Allow',
        Principal: {
          Service: 'ec2.amazonaws.com'
        },
        Action: 'sts:AssumeRole'
      }
    ]
  }

  # Allow the role to take all actions within
  # Amazon Simple Storage Service (Amazon S3)
  # and Amazon DynamoDB across the AWS account.
  policy_arns = [
    'arn:aws:iam::aws:policy/AmazonS3FullAccess',
    'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess'
  ]

  iam_client = Aws::IAM::Client.new

  puts "Attempting to create the role named '#{role_name}'..."

  role_arn = create_role(
    iam_client,
    role_name,
    assume_role_policy_document,
    policy_arns
  )

  if role_arn == 'Error'
    puts 'Could not create role.'
  else
    puts "Role created with ARN '#{role_arn}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Gestione degli utenti IAM

Un utente IAM rappresenta una persona o un servizio con cui interagisce. AWS Per ulteriori informazioni sugli utenti IAM, consulta Utenti IAM.

In questo esempio, usi l'AWSSDK for Ruby con IAM per:

1. Ottieni informazioni sugli utenti AWS IAM disponibili utilizzando Aws: :IAM: :Client #list_users.
2. Crea un utente utilizzando Aws: :IAM: :Client #create_user.
3. Aggiorna il nome dell'utente utilizzando Aws: :IAM: :Client #update_user.
4. Elimina l'utente utilizzando Aws: :IAM: :Client #delete_user.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

Esempio

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

# The following code example shows how to to:
# 1. Get a list of user names in AWS Identity and Access Management (IAM).
# 2. Create a user.
# 3. Update the user's name.
# 4. Delete the user.

require 'aws-sdk-iam'

# Gets a list of available user names in
# AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   list_user_names(Aws::IAM::Client.new)
def list_user_names(iam_client)
  response = iam_client.list_users
  if response.key?('users') && response.users.count.positive?
    response.users.each do |user|
      puts user.user_name
    end
  else
    puts 'No users found.'
```

```
  end
rescue StandardError => e
  puts "Error listing user names: #{e.message}"
end

# Creates a user in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the new user.
# @return [Boolean] true if the user was created; otherwise, false.
# @example
#   exit 1 unless user_created?(Aws::IAM::Client.new, 'my-user')
def user_created?(iam_client, user_name)
  iam_client.create_user(user_name: user_name)
  return true
rescue Aws::IAM::Errors::EntityAlreadyExists
  puts "Error creating user: user '#{user_name}' already exists."
  return false
rescue StandardError => e
  puts "Error creating user: #{e.message}"
  return false
end

# Changes the name of a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_current_name [String] The current name of the user.
# @param user_new_name [String] The new name for the user.
# @return [Boolean] true if the name of the user was changed;
#   otherwise, false.
# @example
#   exit 1 unless user_name_changed?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'my-changed-user'
#   )
def user_name_changed?(iam_client, user_current_name, user_new_name)
  iam_client.update_user(
    user_name: user_current_name,
    new_user_name: user_new_name
  )
```

```
  return true
rescue StandardError => e
  puts "Error updating user name: #{e.message}"
  return false
end

# Deletes a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @return [Boolean] true if the user was deleted; otherwise, false.
# @example
#   exit 1 unless user_deleted?(Aws::IAM::Client.new, 'my-user')
def user_deleted?(iam_client, user_name)
  iam_client.delete_user(user_name: user_name)
  return true
rescue StandardError => e
  puts "Error deleting user: #{e.message}"
  return false
end

# Full example call:
def run_me
  user_name = 'my-user'
  user_changed_name = 'my-changed-user'
  delete_user = true
  iam_client = Aws::IAM::Client.new

  puts "Initial user names are:\n\n"
  list_user_names(iam_client)

  puts "\nAttempting to create user '#{user_name}'..."

  if user_created?(iam_client, user_name)
    puts 'User created.'
  else
    puts 'Could not create user. Stopping program.'
    exit 1
  end

  puts "User names now are:\n\n"
```

```
  list_user_names(iam_client)

  puts "\nAttempting to change the name of the user '#{user_name}' " \
    "to '#{user_changed_name}'..."

  if user_name_changed?(iam_client, user_name, user_changed_name)
    puts 'User name changed.'
    puts "User names now are:\n\n"
    list_user_names(iam_client)

    if delete_user
      # Delete user with changed name.
      puts "\nAttempting to delete user '#{user_changed_name}'..."

      if user_deleted?(iam_client, user_changed_name)
        puts 'User deleted.'
      else
        puts 'Could not delete user. You must delete the user yourself.'
      end

      puts "User names now are:\n\n"
      list_user_names(iam_client)
    end
  else
    puts 'Could not change user name.'
    puts "User names now are:\n\n"
    list_user_names(iam_client)

    if delete_user
      # Delete user with initial name.
      puts "\nAttempting to delete user '#{user_name}'..."

      if user_deleted?(iam_client, user_name)
        puts 'User deleted.'
      else
        puts 'Could not delete user. You must delete the user yourself.'
      end

      puts "User names now are:\n\n"
      list_user_names(iam_client)
    end
  end
end
```

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

Lavorare con le policy IAM;

Una policy IAM è un documento che specifica una o più autorizzazioni. Per ulteriori informazioni sulle politiche IAM, consulta Panoramica delle politiche IAM.

In questo esempio, usi l'AWSSDK for Ruby con IAM per:

1. Crea una policy utilizzando Aws: :IAM: :Client #create_policy.
2. Ottieni informazioni sulla politica, utilizzando Aws: :IAM: :Client #get_policy.
3. Allega la policy a un ruolo, usando Aws: :IAM: :Client #attach_role_policy.
4. Elenca le politiche allegate al ruolo, utilizzando Aws: :IAM: :Client #list_attached_role_policies.
5. Scollega la policy dal ruolo, usando Aws: :IAM: :Client #detach_role_policy.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

Dovrai anche creare il ruolo (my-role) specificato nello script. Puoi farlo nella console IAM.

Esempio

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

# The following code example shows how to:
# 1. Create a policy in AWS Identity and Access Management (IAM).
# 2. Attach the policy to a role.
# 3. List the policies that are attached to the role.
# 4. Detach the policy from the role.

require 'aws-sdk-iam'

# Creates a policy in AWS Identity and Access Management (IAM).
#
```

```
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param policy_name [String] A name for the policy.
# @param policy_document [Hash] The policy definition.
# @return [String] The new policy's Amazon Resource Name (ARN);
#   otherwise, the string 'Error'.
# @example
#   puts create_policy(
#     Aws::IAM::Client.new,
#     'my-policy',
#     {
#       'Version': '2012-10-17',
#       'Statement': [
#         {
#           'Effect': 'Allow',
#           'Action': 's3:ListAllMyBuckets',
#           'Resource': 'arn:aws:s3:::*'
#         }
#       ]
#     }
#   )
def create_policy(iam_client, policy_name, policy_document)
  response = iam_client.create_policy(
    policy_name: policy_name,
    policy_document: policy_document.to_json
  )
  return response.policy.arn
rescue StandardError => e
  puts "Error creating policy: #{e.message}"
  return 'Error'
end

# Attaches a policy to a role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to attach the policy to.
# @param policy_arn [String] The policy's Amazon Resource Name (ARN).
# @return [Boolean] True if the policy was attached to the role;
#   otherwise, false.
# @example
#   exit 1 unless policy_attached_to_role?(
#     Aws::IAM::Client.new,
```

```
#     'my-role',
#     'arn:aws:iam::111111111111:policy/my-policy'
#   )
def policy_attached_to_role?(iam_client, role_name, policy_arn)
  iam_client.attach_role_policy(role_name: role_name, policy_arn: policy_arn)
  return true
rescue StandardError => e
  puts "Error attaching policy to role: #{e.message}"
  return false
end

# Displays a list of policy Amazon Resource Names (ARNs) that are attached to a
# role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role.
# @example
#   list_policy_arns_attached_to_role(Aws::IAM::Client.new, 'my-role')
def list_policy_arns_attached_to_role(iam_client, role_name)
  response = iam_client.list_attached_role_policies(role_name: role_name)
  if response.key?('attached_policies') && response.attached_policies.count.positive?
    response.attached_policies.each do |attached_policy|
      puts "  #{attached_policy.policy_arn}"
    end
  else
    puts 'No policies attached to role.'
  end
rescue StandardError => e
  puts "Error checking for policies attached to role: #{e.message}"
end

# Detaches a policy from a role in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - An existing role with an attached policy.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to detach the policy from.
# @param policy_arn [String] The policy's Amazon Resource Name (ARN).
# @return [Boolean] True if the policy was detached from the role;
#   otherwise, false.
```

```
# @example
#   exit 1 unless policy_detached_from_role?(
#     Aws::IAM::Client.new,
#     'my-role',
#     'arn:aws:iam::111111111111:policy/my-policy'
#   )
def policy_detached_from_role?(iam_client, role_name, policy_arn)
  iam_client.detach_role_policy(role_name: role_name, policy_arn: policy_arn)
  return true
rescue StandardError => e
  puts "Error detaching policy from role: #{e.message}"
  return false
end

# Full example call:
def run_me
  role_name = 'my-role'
  policy_name = 'my-policy'

  # Allows the caller to get a list of all buckets in
  # Amazon Simple Storage Service (Amazon S3) that are owned by the caller.
  policy_document = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Action': 's3:ListAllMyBuckets',
        'Resource': 'arn:aws:s3:::*'
      }
    ]
  }

  detach_policy_from_role = true
  iam_client = Aws::IAM::Client.new

  puts "Attempting to create policy '#{policy_name}'..."
  policy_arn = create_policy(iam_client, policy_name, policy_document)

  if policy_arn == 'Error'
    puts 'Could not create policy. Stopping program.'
    exit 1
  else
    puts 'Policy created.'
  end
```

```
  puts "Attempting to attach policy '#{policy_name}' " \
    "to role '#{role_name}'..."

  if policy_attached_to_role?(iam_client, role_name, policy_arn)
    puts 'Policy attached.'
  else
    puts 'Could not attach policy to role.'
    detach_policy_from_role = false
  end

  puts "Policy ARNs attached to role '#{role_name}':"
  list_policy_arns_attached_to_role(iam_client, role_name)

  if detach_policy_from_role
    puts "Attempting to detach policy '#{policy_name}' " \
      "from role '#{role_name}'..."

    if policy_detached_from_role?(iam_client, role_name, policy_arn)
      puts 'Policy detached.'
    else
      puts 'Could not detach policy from role. You must detach it yourself.'
    end

  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Gestione delle chiavi di accesso IAM

Gli utenti hanno bisogno delle proprie chiavi di accesso per effettuare chiamate programmatiche AWS dall'AWSSDK for Ruby. A questo scopo, è possibile creare, modificare, visualizzare o ruotare le chiavi di accesso (ID chiavi di accesso e chiave di accesso segreta) per gli utenti IAM. Per impostazione predefinita, quando viene creata una chiave di accesso, il suo stato è Attivo. Pertanto, l'utente può utilizzare la chiave di accesso per le chiamate alle API. Per ulteriori informazioni sulle chiavi di accesso, consulta Managing Access Keys for IAM Users.

In questo esempio, usi l'AWSSDK for Ruby con IAM per:

1. Elenca le chiavi di accesso utente AWS IAM, utilizzando Aws: :IAM: :Client #list_access_keys.
2. Crea una chiave di accesso utilizzando Aws: :IAM: :Client #create_access_key.

3. Determina quando le chiavi di accesso sono state utilizzate l'ultima volta, utilizzando Aws: :IAM: :Client #get_access_key_last_used.
4. Disattiva le chiavi di accesso, utilizzando Aws: :IAM: :Client #update_access_key.
5. Elimina la chiave di accesso utilizzando Aws: :IAM: :Client #delete_access_key.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

Dovrai anche creare l'utente (my-user) specificato nello script. Puoi creare un nuovo utente IAM nella console IAM o a livello di codice, come mostrato in Aggiungere un nuovo utente IAM.

Esempio

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

# This code example demonstrates how to:
# 1. List access keys for a user in AWS Identity and Access Management (IAM).
# 2. Create an access key for a user.
# 3. Determine when a user's access keys were last used.
# 4. Deactivate an access key for a user.
# 5. Delete an access key for a user.

require 'aws-sdk-iam'

# Lists information about access keys for a user in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   puts list_access_keys(Aws::IAM::Client.new, 'my-user')
```

```
def list_access_keys(iam, user_name)
  response = iam.list_access_keys(user_name: user_name)

  if response.access_key_metadata.count.positive?
    puts 'Access key IDs:'
    response.access_key_metadata.each do |key_metadata|
      puts "  #{key_metadata.access_key_id}"
    end
  else
    puts "No access keys found for user '#{user_name}'."
  end
rescue Aws::IAM::Errors::NoSuchEntity
  puts "Error listing access keys: cannot find user '#{user_name}'."
  exit 1
rescue StandardError => e
  puts "Error listing access keys: #{e.message}"
end

# Creates an access key for a user in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @return [Aws::IAM::Types::AccessKey] Information about the new access key;
#   otherwise, the string 'Error'.
# @example
#   puts create_access_key(Aws::IAM::Client.new, 'my-user')
def create_access_key(iam, user_name)
  response = iam.create_access_key(user_name: user_name)
  access_key = response.access_key
  puts 'Access key created:'
  puts "  Access key ID: #{access_key.access_key_id}"
  puts "  Secret access key: #{access_key.secret_access_key}"
  puts 'Keep a record of this information in a secure location. ' \
    'This will be the only time you will be able to view the ' \
    'secret access key.'
  return access_key
rescue Aws::IAM::Errors::LimitExceeded
  puts 'Error creating access key: limit exceeded. Cannot create any more. ' \
    'To create more, delete an existing access key, and then try again.'
  return 'Error'
rescue StandardError => e
```

```
  puts "Error creating access key: #{e.message}"
  return 'Error'
end

# Lists information about when access keys for a user in
# AWS Identity and Access Management (IAM) were last used.
#
# Prerequisites:
# - The user in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @example
#   puts access_keys_last_used(Aws::IAM::Client.new, 'my-user')
def access_keys_last_used(iam, user_name)
  response = iam.list_access_keys(user_name: user_name)

  response.access_key_metadata.each do |key_metadata|
    last_used = iam.get_access_key_last_used(access_key_id: key_metadata.access_key_id)
    if last_used.access_key_last_used.last_used_date.nil?
      puts "  Key '#{key_metadata.access_key_id}' not used or date undetermined."
    else
      puts "  Key '#{key_metadata.access_key_id}' last used on " \
      "#{last_used.access_key_last_used.last_used_date}"
    end
  end
rescue StandardError => e
  puts "Error determining when access keys were last used: #{e.message}"
end

# Deactivates an access key in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - A user in IAM.
# - An access key for that user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID of the access key.
# @return [Boolean] true if the access key was deactivated;
#   otherwise, false.
# @example
#   exit 1 unless access_key_deactivated?(
#     Aws::IAM::Client.new,
```

```
#     'my-user',
#     'AKIAIOSFODNN7EXAMPLE'
#   )
def access_key_deactivated?(iam, user_name, access_key_id)
  iam.update_access_key(
    user_name: user_name,
    access_key_id: access_key_id,
    status: 'Inactive'
  )
  return true
rescue StandardError => e
  puts "Error deactivating access key: #{e.message}"
  return false
end

# Deletes an access key in AWS Identity and Access Management (IAM).
#
# Prerequisites:
# - A user in IAM.
# - An access key for that user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID of the access key.
# @return [Boolean] true if the access key was deleted;
#   otherwise, false.
# @example
#   exit 1 unless access_key_deleted?(
#     Aws::IAM::Client.new,
#     'my-user',
#     'AKIAIOSFODNN7EXAMPLE'
#   )
def access_key_deleted?(iam, user_name, access_key_id)
  iam.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  return true
rescue StandardError => e
  puts "Error deleting access key: #{e.message}"
  return false
end

# Full example call:
```

```
def run_me
  iam = Aws::IAM::Client.new
  user_name = 'my-user'
  create_key = true  # Set to false to not create a new access key.
  delete_key = true  # Set to false to not delete any generated access key.

  puts "Access keys for user '#{user_name}' before attempting to create an " \
    'additional access key for the user:'
  list_access_keys(iam, user_name)

  access_key = ''

  if create_key
    puts 'Attempting to create an additional access key...'
    access_key = create_access_key(iam, user_name)

    if access_key == 'Error'
      puts 'Additional access key not created. Stopping program.'
      exit 1
    end

    puts 'Additional access key created. Access keys for user now are:'
    list_access_keys(iam, user_name)
  end

  puts 'Determining when current access keys were last used...'
  access_keys_last_used(iam, user_name)

  if create_key && delete_key
    puts 'Attempting to deactivate additional access key...'

    if access_key_deactivated?(iam, user_name, access_key.access_key_id)
      puts 'Access key deactivated. Access keys for user now are:'
      list_access_keys(iam, user_name)
    else
      puts 'Access key not deactivated. Stopping program.'
      puts 'You will need to delete the access key yourself.'
    end

    puts 'Attempting to delete additional access key...'

    if access_key_deleted?(iam, user_name, access_key.access_key_id)
      puts 'Access key deleted. Access keys for user now are:'
      list_access_keys(iam, user_name)
```

```
    else
      puts 'Access key not deleted. You will need to delete the ' \
        'access key yourself.'
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Utilizzo dei certificati server IAM;

Per abilitare le connessioni HTTPS al tuo sito Web o alla tua applicazioneAWS, devi disporre di un certificato del server SSL/TLS. Per utilizzare un certificato ottenuto da un provider esterno con il tuo sito Web o la tua applicazioneAWS, devi caricare il certificato su IAM o importarlo in AWS Certificate Manager. Per ulteriori informazioni sui certificati server, consulta Lavorare con i certificati server.

In questo esempio, usi l'AWSSDK for Ruby con IAM per:

1. Aggiorna un certificato del server, utilizzando Aws: :IAM: :Client #update_server_certificate.
2. Elimina il certificato del server, utilizzando Aws: :IAM: :Client #delete_server_certificate.
3. Elenca le informazioni su eventuali certificati server rimanenti, utilizzando Aws: :IAM: :Client #list_server_certificates.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

> **Note**
>
> Il certificato del server deve già esistere o lo script genererà un errore Aws: :IAM: :Errors::. NoSuchEntity

Esempio

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

# The following code example shows how to:
# 1. Update a server certificate in AWS Identity and Access Management (IAM).
# 2. List the names of available server certificates.
# 3. Delete a server certificate.

require 'aws-sdk-iam'

# Gets a list of available server certificate names in
# AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @example
#   list_server_certificate_names(Aws::IAM::Client.new)
def list_server_certificate_names(iam_client)
  response = iam_client.list_server_certificates

  if response.key?('server_certificate_metadata_list') &&
    response.server_certificate_metadata_list.count.positive?

    response.server_certificate_metadata_list.each do |certificate_metadata|
      puts certificate_metadata.server_certificate_name
    end
  else
    puts 'No server certificates found. Stopping program.'
    exit 1
  end
rescue StandardError => e
  puts "Error getting server certificate names: #{e.message}"
end

# Changes the name of a server certificate in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
#
# - The server certificate in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param server_certificate_current_name [String] The current name of
```

```
#   the server certificate.
# @param server_certificate_new_name [String] The new name for the
#   the server certificate.
# @return [Boolean] true if the name of the server certificate
#   was changed; otherwise, false.
# @example
#   exit 1 unless server_certificate_name_changed?(
#     Aws::IAM::Client.new,
#     'my-server-certificate',
#     'my-changed-server-certificate'
#   )
def server_certificate_name_changed?(
  iam_client,
  server_certificate_current_name,
  server_certificate_new_name
)
  iam_client.update_server_certificate(
    server_certificate_name: server_certificate_current_name,
    new_server_certificate_name: server_certificate_new_name
  )
  return true
rescue StandardError => e
  puts "Error updating server certificate name: #{e.message}"
  return false
end

# Deletes a server certificate in
# AWS Identity and Access Management (IAM).
#
# Prerequisites:
#
# - The server certificate in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param server_certificate_name [String] The name of the server certificate.
# @return [Boolean] true if the server certificate was deleted;
#   otherwise, false.
# @example
#   exit 1 unless server certificate_deleted?(
#     Aws::IAM::Client.new,
#     'my-server-certificate'
#   )
def server_certificate_deleted?(iam_client, server_certificate_name)
  iam_client.delete_server_certificate(
```

```
    server_certificate_name: server_certificate_name
  )
  return true
rescue StandardError => e
  puts "Error deleting server certificate: #{e.message}"
  return false
end

# Full example call:
def run_me
  server_certificate_name = 'my-server-certificate'
  server_certificate_changed_name = 'my-changed-server-certificate'
  delete_server_certificate = true
  iam_client = Aws::IAM::Client.new

  puts "Initial server certificate names are:\n\n"
  list_server_certificate_names(iam_client)

  puts "\nAttempting to change name of server certificate " \
    " '#{server_certificate_name}' " \
    "to '#{server_certificate_changed_name}'..."

  if server_certificate_name_changed?(
    iam_client,
    server_certificate_name,
    server_certificate_changed_name
  )
    puts 'Server certificate name changed.'
    puts "Server certificate names now are:\n\n"
    list_server_certificate_names(iam_client)

    if delete_server_certificate
      # Delete server certificate with changed name.
      puts "\nAttempting to delete server certificate " \
        "'#{server_certificate_changed_name}'..."

      if server_certificate_deleted?(iam_client, server_certificate_changed_name)
        puts 'Server certificate deleted.'
      else
        puts 'Could not delete server certificate. You must delete it yourself.'
      end

      puts "Server certificate names now are:\n\n"
      list_server_certificate_names(iam_client)
```

```
    end
  else
    puts 'Could not change server certificate name.'
    puts "Server certificate names now are:\n\n"
    list_server_certificate_names(iam_client)

    if delete_server_certificate
      # Delete server certificate with initial name.
      puts "\nAttempting to delete server certificate '#{server_certificate_name}'..."

      if server_certificate_deleted?(iam_client, server_certificate_name)
        puts 'Server certificate deleted.'
      else
        puts 'Could not delete server certificate. You must delete it yourself.'
      end

      puts "Server certificate names now are:\n\n"
      list_server_certificate_names(iam_client)
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Gestire gli alias per l'account IAM

Se desideri che l'URL della tua pagina di accesso contenga il nome della tua azienda o un altro identificativo descrittivo anziché l'ID dell'AWSaccount, puoi creare un alias dell'account IAM per l'ID del tuo account. AWS Se crei un alias per un account IAM, l'URL della pagina di accesso cambia per incorporare l'alias. Per ulteriori informazioni sugli alias degli account IAM, consulta Your AWSAccount ID e Its Alias.

In questo esempio, usi l'AWSSDK for Ruby con IAM per:

1. Elenca gli alias degli AWS account utilizzando Aws: :IAM: :Client #list_account_aliases.
2. Crea un alias di account utilizzando Aws: :IAM: :Client #create_account_alias.
3. Elimina l'alias dell'account utilizzando Aws: :IAM: :Client #delete_account_alias.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

Nel codice di esempio, modifica la my-account-aliasstringa in qualcosa che sarà unico per tutti i prodotti Amazon Web Services.

Esempio

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

# The following code example shows how to:
# 1. List available AWS account aliases.
# 2. Create an account alias.
# 3. Delete an account alias.

require 'aws-sdk-iam'

# Lists available AWS account aliases.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @example
#   puts list_aliases(Aws::IAM::Client.new)
def list_aliases(iam)
  response = iam.list_account_aliases

  if response.account_aliases.count.positive?
    response.account_aliases.each do |account_alias|
      puts "  #{account_alias}"
    end
  else
    puts 'No account aliases found.'
  end
rescue StandardError => e
  puts "Error listing account aliases: #{e.message}"
end
```

```
# Creates an AWS account alias.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
# @example
#   exit 1 unless alias_created?(Aws::IAM::Client.new, 'my-account-alias')
def alias_created?(iam, account_alias)
  iam.create_account_alias(account_alias: account_alias)
  return true
rescue StandardError => e
  puts "Error creating account alias: #{e.message}"
  return false
end

# Deletes an AWS account alias.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param account_alias [String] The name of the account alias to delete.
# @return [Boolean] true if the account alias was deleted; otherwise, false.
# @example
#   exit 1 unless alias_deleted?(Aws::IAM::Client.new, 'my-account-alias')
def alias_deleted?(iam, account_alias)
  iam.delete_account_alias(account_alias: account_alias)
  return true
rescue StandardError => e
  puts "Error deleting account alias: #{e.message}"
  return false
end

# Full example call:
def run_me
  iam = Aws::IAM::Client.new
  account_alias = 'my-account-alias'
  create_alias = true # Change to false to not generate an account alias.
  delete_alias = true # Change to false to not delete any generated account alias.

  puts 'Account aliases are:'
  list_aliases(iam)

  if create_alias
    puts 'Attempting to create account alias...'
    if alias_created?(iam, account_alias)
      puts 'Account alias created. Account aliases now are:'
```

```
      list_aliases(iam)
    else
      puts 'Account alias not created. Stopping program.'
      exit 1
    end
  end

  if create_alias && delete_alias
    puts 'Attempting to delete account alias...'
    if alias_deleted?(iam, account_alias)
      puts 'Account alias deleted. Account aliases now are:'
      list_aliases(iam)
    else
      puts 'Account alias not deleted. You will need to delete ' \
        'the alias yourself.'
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

AWS Key Management ServiceEsempi di utilizzo dell'AWSSDK for Ruby

AWS Key Management Service(AWS KMS) è un servizio di crittografia e gestione delle chiavi scalato per il cloud. È possibile utilizzare i seguenti esempi per accedere AWS KMS utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su AWS KMS, consulta la documentazione su AWS KMS. Per informazioni di riferimento sul AWS KMS client, vedi Aws: :KMS: :Client.

Argomenti

- Creazione di un AWS KMS key
- Crittografia dei dati in AWS KMS
- Decrittografia di un data blob in AWS KMS
- Ricrittografia di un data blob in AWS KMS

Creazione di un AWS KMS key

L'esempio seguente utilizza il metodo AWS SDK for Ruby create_key, che CreateKeyimplementa l'operazione per creare un. AWS KMS keys Poiché l'esempio crittografa solo una piccola quantità di

dati, una chiave KMS va bene per i nostri scopi. Per grandi quantità di dati, usa la chiave KMS per crittografare una chiave di crittografia dei dati (DEK).

```
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
```

Vedi l'esempio completo su. GitHub

Crittografia dei dati in AWS KMS

L'esempio seguente utilizza il metodo di crittografia AWS SDK for Ruby, che implementa l'operazione Encrypt, per crittografare la stringa «1234567890». L'esempio visualizza una versione leggibile del blob crittografato risultante.

```
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*")
```

Vedi l'esempio completo su. GitHub

Decrittografia di un data blob in AWS KMS

L'esempio seguente utilizza il metodo di decrittografia AWS SDK for Ruby, che implementa l'operazione Decrypt, per decrittografare la stringa fornita ed emettere il risultato.

```
require "aws-sdk-kms" # v2: require 'aws-sdk'

# Decrypted blob

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09
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
```

GitHubVedi l'esempio completo su.

Ricrittografia di un data blob in AWS KMS

L'esempio seguente utilizza il metodo AWS SDK for Ruby re_encrypt, che ReEncryptimplementa l'operazione, per decrittografare i dati crittografati e quindi ricrittografare immediatamente i dati con uno nuovo. AWS KMS key Le operazioni vengono eseguite interamente su lato server all'interno

di AWS KMS, in modo che il testo normale non dovrà mai essere esposto all'esterno di AWS KMS. L'esempio visualizza una versione leggibile del blob ricrittografato risultante.

```
require "aws-sdk-kms"  # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09
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*")
```

Vedi l'esempio completo su. GitHub

AWS LambdaEsempi di utilizzo dell'AWSSDK for Ruby

AWS Lambda(Lambda) è una piattaforma di elaborazione senza amministrazione per sviluppatori web di backend che esegue il codice per te nel AWS cloud e ti offre una struttura dei prezzi dettagliata. È possibile utilizzare i seguenti esempi per accedere a Lambda utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Lambda, consulta la AWS Lambda documentazione.

Argomenti

- Visualizzazione delle informazioni su tutte le funzioni Lambda
- Creazione di una funzione Lambda
- Esecuzione di una funzione Lambda

- Configurazione di una funzione Lambda per ricevere notifiche

Visualizzazione delle informazioni su tutte le funzioni Lambda

L'esempio seguente mostra il nome, l'ARN e il ruolo di tutte le funzioni Lambda nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

client.list_functions.functions.each do |function|
  puts 'Name: ' + function.function_name
  puts 'ARN:  ' + function.function_arn
  puts 'Role: ' + function.role
  puts
end
```

Creazione di una funzione Lambda

L'esempio seguente crea la funzione Lambda denominata `my-notification-function` nella `us-west-2` regione utilizzando questi valori:

- Ruolo ARN:. `my-resource-arn` Nella maggior parte dei casi, è necessario allegare solo la politica `AWSLambdaExecute` gestita alla politica per questo ruolo.
- Punto di ingresso della funzione: `my-package.my-class`
- Runtime: `java8`
- File zip: `my-zip-file.zip`

- Secchio: `my-notification-bucket`
- Chiave: `my-zip-file`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

args = {}
args[:role] = 'my-resource-arn'
args[:function_name] = 'my-notification-function'
args[:handler] = 'my-package.my-class'

# Also accepts nodejs, nodejs4.3, and python2.7
args[:runtime] = 'java8'

code = {}
code[:zip_file] = 'my-zip-file.zip'
code[:s3_bucket] = 'my-notification-bucket'
code[:s3_key] = 'my-zip-file'

args[:code] = code

client.create_function(args)
```

Esecuzione di una funzione Lambda

L'esempio seguente esegue la funzione Lambda denominata `MyGetitemsFunction` nella `us-west-2` regione. Questa funzione restituisce un elenco di elementi da un database. L'input JSON è simile al seguente.

```
{
   "SortBy": "name|time",
   "SortOrder": "ascending|descending",
   "Number": 50
}
```

dove:

- `SortBy`è il criterio per l'ordinamento dei risultati. Il nostro esempio utilizza`time`, il che significa che gli articoli restituiti vengono ordinati nell'ordine in cui sono stati aggiunti al database.
- `SortOrder`è l'ordine di ordinamento. Il nostro esempio utilizza`descending`, il che significa che l'elemento più recente è l'ultimo nell'elenco.
- `Number`è il numero massimo di elementi da recuperare (il valore predefinito è 50). Il nostro esempio utilizza`10`, il che significa che ottieni i 10 elementi più recenti.

L'output JSON è simile al seguente, dove:

- `STATUS-CODE`è un codice di stato HTTP, `200` significa che la chiamata ha avuto successo.
- `RESULT`è il risultato della chiamata, `success` oppure`failure`.
- `ERROR`è un messaggio di errore se `result` è`failure`, altrimenti una stringa vuota
- `DATA`è un array di risultati restituiti se `result` è`success`, altrimenti nullo.

```
{
   "statusCode": "STATUS-CODE",
   "body": {
      "result": "RESULT",
      "error": "ERROR",
      "data": "DATA"
   }
}
```

Il primo passo è caricare i moduli che utilizziamo:

- `aws-sdk`carica il modulo AWS SDK for Ruby che utilizziamo per richiamare la funzione Lambda.
- `json`carica il modulo JSON che utilizziamo per marshall e unmarshall i payload di richiesta e risposta.

- oscarica il modulo del sistema operativo che utilizziamo per assicurarci di poter eseguire la nostra applicazione Ruby su Microsoft Windows. Se utilizzi un sistema operativo diverso, puoi rimuovere quelle righe.
- Quindi creiamo il client Lambda che utilizziamo per richiamare la funzione Lambda.
- Successivamente creiamo l'hash per gli argomenti della richiesta e chiamiamo. `MyGetItemsFunction`
- Infine analizziamo la risposta e, se ha esito positivo, stampiamo gli elementi.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'
require 'json'

# To run on Windows:
require 'os'
if OS.windows?
  Aws.use_bundled_cert!
end

client = Aws::Lambda::Client.new(region: 'us-west-2')

# Get the 10 most recent items
req_payload = {:SortBy => 'time', :SortOrder => 'descending', :NumberToGet => 10}
payload = JSON.generate(req_payload)

resp = client.invoke({
                       function_name: 'MyGetItemsFunction',
                       invocation_type: 'RequestResponse',
                       log_type: 'None',
                       payload: payload
                     })
```

```
resp_payload = JSON.parse(resp.payload.string) # , symbolize_names: true)

# If the status code is 200, the call succeeded
if resp_payload["statusCode"] == 200
  # If the result is success, we got our items
  if resp_payload["body"]["result"] == "success"
    # Print out items
    resp_payload["body"]["data"].each do |item|
      puts item
    end
  end
end
```

Vedi l'esempio completo su. GitHub

Configurazione di una funzione Lambda per ricevere notifiche

L'esempio seguente configura la funzione Lambda `my-notification-function` denominata `us-west-2` nella regione per accettare notifiche dalla risorsa con l'ARN. `my-resource-arn`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-lambda'  # v2: require 'aws-sdk'

client = Aws::Lambda::Client.new(region: 'us-west-2')

args = {}
args[:function_name] = 'my-notification-function'
args[:statement_id] = 'lambda_s3_notification'
args[:action] = 'lambda:InvokeFunction'
args[:principal] = 's3.amazonaws.com'
args[:source_arn] = 'my-resource-arn'
```

```
client.add_permission(args)
```

Esempi di Amazon Polly con l'AWSSDK for Ruby

Amazon Polly è un servizio cloud che converte il testo in voce naturale. Gli esempi di AWS SDK for Ruby possono integrare Amazon Polly nelle tue applicazioni. Per ulteriori informazioni su Amazon Polly, consulta la documentazione di Amazon Polly. Gli esempi presuppongono che tu abbia già impostato e configurato l'SDK (ovvero che tu abbia importato tutti i pacchetti richiesti e impostato le credenziali e la regione). Per ulteriori informazioni, consulta Installazione dell'AWSSDK per Ruby e AWS Configurazione dell'SDK per Ruby.

Argomenti

- Ottenere un elenco di voci
- Ottenere un elenco di lessici
- Sintetizzazione del parlato

Ottenere un elenco di voci

Questo esempio utilizza il metodo describe_voices per ottenere l'elenco delle voci in inglese americano nella regione. `us-west-2`

Scegliete `Copy` di salvare il codice localmente.

Crea il file polly_describe_voices.rb.

Aggiungi la gemma richiesta.

> **Note**
>
> La versione 2 dell'AWSSDK for Ruby non conteneva gemme specifiche del servizio.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
```

```
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts '  ' + v.gender
    puts
  end
rescue StandardError => ex
  puts 'Could not get voices'
  puts 'Error message:'
  puts ex.message
end
```

Vedi l'esempio completo su. GitHub

Ottenere un elenco di lessici

Questo esempio utilizza il metodo list_lexicons per ottenere l'elenco dei lessici nella regione. `us-west-2`

Scegliete `Copy` di salvare il codice localmente.

Crea il file polly_list_lexicons.rb.

Aggiungi la gemma richiesta.

> **Note**
>
> La versione 2 dell'AWSSDK for Ruby non conteneva gemme specifiche del servizio.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts '  Alphabet:' + l.attributes.alphabet
    puts '  Language:' + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts ex.message
end
```

Vedi l'esempio completo su. GitHub

Sintetizzazione del parlato

Questo esempio utilizza il metodo synthesize_speech per ottenere il testo da un file e produrre un file MP3 contenente il parlato sintetizzato.

Scegliete Copy di salvare il codice localmente.

Crea il file polly_synthesize_speech.rb.

Aggiungi la gemma richiesta.

> **Note**
>
> La versione 2 dell'AWSSDK for Ruby non conteneva gemme specifiche del servizio.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-polly'  # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?()
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
```

```
  else
    puts 'No such file: ' + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = first_part + '.mp3'

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts 'Wrote MP3 content to: ' + mp3_file
rescue StandardError => ex
  puts 'Got error:'
  puts 'Error message:'
  puts ex.message
end
```

> **Note**
>
> Il file MP3 risultante è in formato MPEG-2.

Vedi l'esempio completo su. GitHub

Esempi di Amazon RDS con l'AWSSDK for Ruby

Amazon Relational Database Service (Amazon RDS) è un servizio Web che semplifica la configurazione, il funzionamento e la scalabilità di un database relazionale nel cloud. Puoi utilizzare i seguenti esempi per accedere ad Amazon RDS utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon RDS, consulta la documentazione di Amazon Relational Database Service.

> **Note**
>
> Alcuni dei seguenti esempi utilizzano metodi introdotti nella `2.2.18` versione della `Aws::RDS::Resource` classe. Per eseguire questi esempi, è necessario utilizzare quella versione o una versione successiva della `aws-sdk` gemma.

Argomenti

Ottenere informazioni su tutte le istanze Amazon RDS

L'esempio seguente elenca il nome (ID) e lo stato di tutte le istanze Amazon RDS nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
```

```
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_instances.each do |i|
  puts "Name (ID): #{i.id}"
  puts "Status   : #{i.db_instance_status}"
  puts
end
```

Ottenere informazioni su tutte le istantanee di Amazon RDS

L'esempio seguente elenca i nomi (ID) e lo stato di tutti gli snapshot Amazon RDS (istanza) nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_snapshots.each do |s|
  puts "Name (ID): #{s.snapshot_id}"
  puts "Status:    #{s.status}"
end
```

Ottenere informazioni su tutti i cluster Amazon RDS e i relativi snapshot

L'esempio seguente elenca il nome (ID) e lo stato di tutti i cluster Amazon RDS e il nome (ID) e lo stato dei relativi snapshot nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_clusters.each do |c|
  puts "Name (ID): #{c.id}"
  puts "Status:    #{c.status}"

  c.snapshots.each do |s|
    puts "  Snapshot: #{s.snapshot_id}"
    puts "  Status:   #{s.status}"
  end
end
```

Ottenere informazioni su tutti i gruppi di sicurezza Amazon RDS

L'esempio seguente elenca i nomi di tutti i gruppi di sicurezza Amazon RDS nella regione. `us-west-2`

> **Note**
>
> I gruppi di sicurezza Amazon RDS sono applicabili solo quando utilizzi la piattaforma Amazon EC2-Classic. Se utilizzi Amazon EC2-VPC, usa i gruppi di sicurezza VPC. Entrambi sono mostrati nell'esempio.

> **Warning**
>
> Ritireremo EC2-Classic il 15 agosto 2022. Suggeriamo di effettuare la migrazione da EC2-Classic a un VPC. Per ulteriori informazioni, consulta Eseguire la migrazione da EC2-Classic a un VPC nella Guida per l'utente di Amazon EC2 per istanze Linux o nella Guida per l'utente di Amazon EC2 per istanze Windows. Per ulteriori informazioni, consulta il post di blog Il networking EC2-Classic va in pensione: ecco come prepararsi.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_instances.each do |i|
  # Show any security group IDs and descriptions
  puts 'Security Groups:'

  i.db_security_groups.each do |sg|
    puts sg.db_security_group_name
    puts '  ' + sg.db_security_group_description
    puts
  end

  # Show any VPC security group IDs and their status
  puts 'VPC Security Groups:'

  i.vpc_security_groups.each do |vsg|
    puts vsg.vpc_security_group_id
    puts '  ' + vsg.status
    puts
```

```
  end
end
```

Ottenere informazioni su tutti i gruppi di sottoreti Amazon RDS

L'esempio seguente elenca il nome e lo stato di tutti i tuoi sottogruppi di sottoreti Amazon RDS nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_subnet_groups.each do |s|
  puts s.name
  puts '  ' + s.subnet_group_status
end
```

Ottenere informazioni su tutti i gruppi di parametri Amazon RDS

L'esempio seguente elenca i nomi e le descrizioni di tutti i gruppi di parametri Amazon RDS nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
```

```
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_parameter_groups.each do |p|
  puts p.db_parameter_group_name
  puts '  ' + p.description
end
```

Creazione di uno snapshot di un'istanza Amazon RDS

L'esempio seguente crea uno snapshot per l'istanza Amazon RDS rappresentata da instance_name nella regione. `us-west-2`

> **Note**
>
> Se la tua istanza è membro di un cluster, non puoi creare uno snapshot dell'istanza. È invece necessario creare uno snapshot del cluster (vedi Creazione di uno snapshot di un cluster Amazon RDS).

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

instance = rds.db_instance(instance_name)
```

```
date =  Time.new
date_time =  date.year.to_s +  '-' +  date.month.to_s +  '-' +  date.day.to_s +  '-' +
 date.hour.to_s +  '-' +  date.min.to_s

id = instance_name + '-' + date_time

instance.create_snapshot({db_snapshot_identifier: id})

puts "Created snapshot #{id}"
```

Creazione di uno snapshot di un cluster Amazon RDS

L'esempio seguente crea uno snapshot per il cluster Amazon RDS rappresentato da cluster_name nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-rds'  # v2: require 'aws-sdk'

rds = Aws::RDS::Resource.new(region: 'us-west-2')

cluster = rds.db_cluster(cluster_name)

date =  Time.new
date_time =  date.year.to_s +  '-' +  date.month.to_s +  '-' +  date.day.to_s +  '-' +
 date.hour.to_s +  '-' +  date.min.to_s

id = cluster_name + '-' + date_time

cluster.create_snapshot({db_cluster_snapshot_identifier: id})

puts "Created cluster snapshot #{id}"
```

Esempi di Amazon SES con l'AWSSDK for Ruby

Amazon Simple Email Service (Amazon SES) È una piattaforma di posta elettronica che offre un modo semplice ed economico per inviare e ricevere e-mail utilizzando i propri indirizzi e-mail e domini. Puoi utilizzare i seguenti esempi per accedere ad Amazon SES utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon SES, consulta la documentazione di Amazon SES.

Argomenti

Pubblicazione di indirizzi e-mail Amazon SES validi

L'esempio seguente mostra come utilizzare l'AWSSDK for Ruby per elencare gli indirizzi e-mail Amazon SES validi.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Create client in us-west-2 region
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})
```

```
ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

Vedi l'esempio completo su. GitHub

Verifica di un indirizzo e-mail in Amazon SES

L'esempio seguente dimostra come utilizzare l'AWSSDK for Ruby per verificare un indirizzo e-mail Amazon SES.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
```

```
  ses.verify_email_identity({
    email_address: recipient
  })

  puts 'Email sent to ' + recipient

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

Vedi l'esempio completo su. GitHub

Invio di un messaggio a un indirizzo e-mail in Amazon SES

L'esempio seguente mostra come utilizzare l'AWSSDK for Ruby per inviare un messaggio a un indirizzo e-mail Amazon SES.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"
```

```
# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
```

```
      }
    },
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts 'Email sent to ' + recipient


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

Vedi l'esempio completo su. GitHub

Ottenere statistiche su Amazon SES

L'esempio seguente mostra come utilizzare l'AWSSDK for Ruby per ottenere statistiche su Amazon SES. Utilizza queste informazioni per evitare di danneggiare la tua reputazione quando le e-mail vengono respinte o rifiutate.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-ses'  # v2: require 'aws-sdk'

# 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')

begin
  # Get send statistics so we don't ruin our reputation
```

```
  resp = ses.get_send_statistics({})

  dps = resp.send_data_points

  puts "Got #{dps.count} data point(s):"
  puts

  dps.each do |dp|
    puts "Timestamp:  #{dp.timestamp}" #=> Time
    puts "Attempts:   #{dp.delivery_attempts}" #=> Integer
    puts "Bounces:    #{dp.bounces}" #=> Integer
    puts "Complaints: #{dp.complaints}" #=> Integer
    puts "Rejects:    #{dp.rejects}"  #-> Integer
    puts
  end

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Error: #{error}"
end
```

Vedi l'esempio completo su. GitHub

Esempi di Amazon SNS con l'AWSSDK for Ruby

Amazon Simple Notification Service (Amazon SNS) è un servizio Web che consente alle applicazioni, agli utenti finali e ai dispositivi di inviare e ricevere istantaneamente notifiche dal cloud. Puoi utilizzare i seguenti esempi per accedere ad Amazon SNS utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon SNS, consulta la documentazione di Amazon SNS.

Argomenti

- Ottenere informazioni su tutti gli argomenti di Amazon SNS
- Creazione di un argomento Amazon SNS
- Ottenere informazioni su tutti gli abbonamenti in un argomento Amazon SNS
- Creazione di un abbonamento in un argomento di Amazon SNS
- Invio di un messaggio a tutti gli abbonati ad Amazon SNS Topic
- Abilitazione della pubblicazione di una risorsa su un argomento Amazon SNS

Ottenere informazioni su tutti gli argomenti di Amazon SNS

L'esempio seguente elenca gli ARN dei tuoi argomenti Amazon SNS nella `us-west-2` regione.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

sns.topics.each do |topic|
  puts topic.arn
end
```

Creazione di un argomento Amazon SNS

L'esempio seguente crea l'argomento `MyGroovyTopic` nella `us-west-2` regione e visualizza l'ARN dell'argomento risultante.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'
```

```
sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.create_topic(name: 'MyGroovyTopic')
puts topic.arn
```

Ottenere informazioni su tutti gli abbonamenti in un argomento Amazon SNS

L'esempio seguente elenca gli indirizzi e-mail degli abbonamenti Amazon SNS per l'argomento con l'`arn:aws:sns:us-west-2:123456789:MyGroovyTopic`ARN della regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.subscriptions.each do |s|
  puts s.attributes['Endpoint']
end
```

Creazione di un abbonamento in un argomento di Amazon SNS

L'esempio seguente crea una sottoscrizione per l'argomento con l'ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic` per un utente che ha l'indirizzo e-mail `MyGroovyUser@MyGroovy.com` nella `us-west-2` regione e visualizza l'ARN risultante. Inizialmente il valore ARN è in attesa di conferma. Quando l'utente conferma il proprio indirizzo e-mail, questo valore diventa un vero ARN.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

sub = topic.subscribe({
  protocol: 'email',
  endpoint: 'MyGroovyUser@MyGroovy.com'
})

puts sub.arn
```

Invio di un messaggio a tutti gli abbonati ad Amazon SNS Topic

L'esempio seguente invia il messaggio «Hello!» a tutti gli abbonati all'argomento Amazon SNS con ARN. `arn:aws:sns:us-west-2:123456789:MyGroovyTopic`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'
```

```
sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.publish({
  message: 'Hello!'
})
```

Abilitazione della pubblicazione di una risorsa su un argomento Amazon SNS

L'esempio seguente consente alla risorsa con l'ARN di `my-resource-arn` pubblicare sull'argomento con l'ARN nella regione. `my-topic-arn` `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sns'  # v2: require 'aws-sdk'

policy  = '{
  "Version":"2008-10-17",
  "Id":"__default_policy_ID",
  "Statement":[{
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
    "Principal":{
      "AWS":"*"
    },
    "Action":["SNS:Publish"],
    "Resource":"' + my-topic-arn + '",
    "Condition":{
      "ArnEquals":{
        "AWS:SourceArn":"' + my-resource-arn + '"}
     }
  }]
```

```
}'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

# Get topic by ARN
topic = sns.topic(my-topic-arn)

# Add policy to topic
topic.set_attributes({
  attribute_name: "Policy",
  attribute_value: policy
})
```

Esempi di Amazon SQS con l'AWSSDK for Ruby

Amazon Simple Queue Service (Amazon SQS) è un servizio di accodamento dei messaggi completamente gestito che semplifica il disaccoppiamento e la scalabilità di microservizi, sistemi distribuiti e applicazioni serverless. Puoi utilizzare i seguenti esempi per accedere ad Amazon SQS utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon SQS, consulta la documentazione di Amazon SQS.

Argomenti

- Ottenere informazioni su tutte le code in Amazon SQS
- Creazione di una coda in Amazon SQS
- Utilizzo delle code in Amazon SQS
- Invio di messaggi in Amazon SQS
- Invio e ricezione di messaggi in Amazon SQS
- Ricezione di messaggi in Amazon SQS
- Ricezione di messaggi tramite Long Polling in Amazon SQS
- Abilitazione del long polling in Amazon SQS
- Ricezione di messaggi utilizzando la QueuePoller classe in Amazon SQS
- Reindirizzamento di lettere morte in Amazon SQS
- Eliminazione di una coda in Amazon SQS
- Abilitazione della pubblicazione di una risorsa in una coda in Amazon SQS
- Utilizzo di una coda di lettere morte in Amazon SQS

- Specificazione del timeout di visibilità dei messaggi in Amazon SQS

Ottenere informazioni su tutte le code in Amazon SQS

L'esempio seguente elenca gli URL, gli ARN, i messaggi disponibili e i messaggi in fuga delle code Amazon SQS nella regione. `us-west-2`

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Lists the URLs of available queues in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-east-1'))
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-east-1'),
#     'https://sqs.us-east-1.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:
def run_me
  region = 'us-east-1'
  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-east-1.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

run_me if $PROGRAM_NAME == __FILE__
```

Creazione di una coda in Amazon SQS

L'esempio seguente crea la coda Amazon SQS denominata `MyGroovyQueue` nella `us-west-2` regione e ne visualizza l'URL.

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

require 'aws-sdk-sqs'

# Creates a queue in Amazon Simple Queue Service (Amazon 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-east-1'),
#     '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:
def run_me
  region = 'us-east-1'
  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

run_me if $PROGRAM_NAME == __FILE__
```

Utilizzo delle code in Amazon SQS

Amazon SQS offre code ospitate altamente scalabili per l'archiviazione dei messaggi mentre viaggiano tra applicazioni o microservizi. Per ulteriori informazioni sulle code, consulta How Amazon SQS Queues Work.

In questo esempio, utilizzi l'AWSSDK for Ruby con Amazon SQS per:

1. Ottieni un elenco delle tue code utilizzando. `Aws::SQS::Client#list_queues`
2. Crea una coda utilizzando. `Aws::SQS::Client#create_queue`

3. Ottieni l'URL della coda utilizzando. `Aws::SQS::Client#get_queue_url`
4. Elimina la coda utilizzando. `Aws::SQS::Client#delete_queue`

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

Esempio

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Get a list of your queues.
# 2. Create a queue.
# 3. Get the queue's URL.
# 4. Delete the queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Get a list of your queues.
sqs.list_queues.queue_urls.each do |queue_url|
  puts queue_url
end

# Create a queue.
```

```
queue_name = "my-queue"

begin
  sqs.create_queue({
    queue_name: queue_name,
    attributes: {
      "DelaySeconds" => "60", # Delay message delivery for 1 minute (60 seconds).
      "MessageRetentionPeriod" => "86400" # Delete message after 1 day (24 hours * 60
 minutes * 60 seconds).
    }
  })
rescue Aws::SQS::Errors::QueueDeletedRecently
  puts "A queue with the name '#{queue_name}' was recently deleted. Wait at least 60
 seconds and try again."
  exit(false)
end

# Get the queue's URL.
queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
puts queue_url

# Delete the queue.
sqs.delete_queue(queue_url: queue_url)
```

Invio di messaggi in Amazon SQS

L'esempio seguente invia il messaggio «Hello world» tramite la coda di Amazon SQS con l'URL URL nella regione. `us-west-2`

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends a message to 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 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-east-1'),
#     'https://sqs.us-east-1.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:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.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

run_me if $PROGRAM_NAME == __FILE__
```

L'esempio seguente invia i messaggi «Hello world» e «How is the weather?» tramite la coda Amazon SQS con l'URL URL nella regione. `us-west-2`

> **Note**
>
> Se la tua coda è una coda FIFO, devi includere un `message_group_id` parametro oltre ai parametri and. `id` `message_body`

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends multiple messages as a batch to 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 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-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
```

```
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
def run_me
  region = 'us-east-1'
  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-east-1.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

run_me if $PROGRAM_NAME == __FILE__
```

Invio e ricezione di messaggi in Amazon SQS

Dopo aver creato una coda in Amazon SQS, puoi inviarle un messaggio e poi consumarla. Per ulteriori informazioni, consulta Tutorial: invio di un messaggio a una coda Amazon SQS e Tutorial: ricezione ed eliminazione di un messaggio da una coda Amazon SQS.

In questo esempio, utilizzi l'AWSSDK for Ruby con Amazon SQS per:

1. Invia un messaggio a una coda utilizzando Aws: :SQS: :Client #send_message.

> **Note**
>
> Se la tua coda è una coda FIFO, devi includere un `message_group_id` parametro oltre ai parametri and. `id` `message_body`

1. Ricevi il messaggio in coda utilizzando Aws: :SQS: :Client #receive_message.
2. Visualizza informazioni sul messaggio.
3. Elimina il messaggio dalla coda utilizzando Aws: :SQS: :Client #delete_message.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

È inoltre necessario creare la coda my-queue, operazione che è possibile eseguire nella console Amazon SQS.

Esempio

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
```

```
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Send a message to a queue.
# 2. Receive the message in the queue.
# 3. Display information about the message.
# 4. Delete the message from the queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Send a message to a queue.
queue_name = "my-queue"

begin
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Create a message with three custom attributes: Title, Author, and WeeksOn.
  send_message_result = sqs.send_message({
    queue_url: queue_url,
    message_body: "Information about current NY Times fiction bestseller for week of
 2016-12-11.",
    message_attributes: {
      "Title" => {
        string_value: "The Whistler",
        data_type: "String"
      },
      "Author" => {
        string_value: "John Grisham",
        data_type: "String"
      },
      "WeeksOn" => {
        string_value: "6",
        data_type: "Number"
      }
    }
  })
rescue Aws::SQS::Errors::NonExistentQueue
```

```
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
end

puts send_message_result.message_id

# Receive the message in the queue.
receive_message_result = sqs.receive_message({
  queue_url: queue_url,
  message_attribute_names: ["All"], # Receive all custom attributes.
  max_number_of_messages: 1, # Receive at most one message.
  wait_time_seconds: 0 # Do not wait to check for the message.
})

# Display information about the message.
# Display the message's body and each custom attribute value.
receive_message_result.messages.each do |message|
  puts message.body
  puts "Title: #{message.message_attributes["Title"]["string_value"]}"
  puts "Author: #{message.message_attributes["Author"]["string_value"]}"
  puts "WeeksOn: #{message.message_attributes["WeeksOn"]["string_value"]}"

  # Delete the message from the queue.
  sqs.delete_message({
    queue_url: queue_url,
    receipt_handle: message.receipt_handle
  })
end
```

Ricezione di messaggi in Amazon SQS

L'esempio seguente mostra il corpo di un massimo di 10 messaggi nella coda Amazon SQS con l'URL URL nella regione. `us-west-2`

> **Note**
>
> `receive_message`non garantisce la ricezione di tutti i messaggi (vedi Proprietà delle code distribuite) e per impostazione predefinita non elimina il messaggio.

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

```
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-east-1'),
#     'https://sqs.us-east-1.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:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.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

run_me if $PROGRAM_NAME == __FILE__
```

Ricezione di messaggi tramite Long Polling in Amazon SQS

L'esempio seguente attende fino a 10 secondi per visualizzare i corpi di un massimo di 10 messaggi nella coda Amazon SQS con l'URLURL nella regione. `us-west-2`

Se non specifichi un tempo di attesa, il valore predefinito è 0 (Amazon SQS non aspetta).

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'
```

```
sqs = Aws::SQS::Client.new(region: 'us-west-2')

resp = sqs.receive_message(queue_url: URL, max_number_of_messages: 10,
 wait_time_seconds: 10)

resp.messages.each do |m|
  puts m.body
end
```

Abilitazione del long polling in Amazon SQS

Il polling prolungato aiuta a ridurre i costi di utilizzo di Amazon SQS riducendo il numero di risposte vuote ed eliminando le false risposte vuote. Per ulteriori informazioni sul long polling, consulta Amazon SQS Long Polling.

In questo esempio, utilizzi l'AWSSDK for Ruby con Amazon SQS per:

1. Crea una coda e impostala per un polling prolungato utilizzando Aws: :SQS: :Client #create_queue.
2. Imposta un polling lungo per una coda esistente utilizzando Aws: :SQS: :Client #set_queue_attributes.
3. Imposta un polling lungo quando ricevi messaggi per una coda utilizzando Aws: :SQS: :Client #receive_message.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

È inoltre necessario creare le code esistenting-queue e receive-queue, operazione che puoi fare nella console Amazon SQS.

Esempio

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
```

```
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Create a queue and set it for long polling.
# 2. Set long polling for an existing queue.
# 3. Set long polling when receiving messages for a queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue and set it for long polling.
new_queue_name = "new-queue"

create_queue_result = sqs.create_queue({
  queue_name: new_queue_name,
  attributes: {
    "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
  },
})

puts create_queue_result.queue_url

# Set long polling for an existing queue.
begin
  existing_queue_name = "existing-queue"
  existing_queue_url = sqs.get_queue_url(queue_name: existing_queue_name).queue_url

  sqs.set_queue_attributes({
    queue_url: existing_queue_url,
    attributes: {
      "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
    },
  })
rescue Aws::SQS::Errors::NonExistentQueue
```

```
  puts "Cannot set long polling for a queue named '#{existing_queue_name}', as it does
 not exist."
end

# Set long polling when receiving messages for a queue.

# 1. Using receive_message.
begin
  receive_queue_name = "receive-queue"
  receive_queue_url = sqs.get_queue_url(queue_name: receive_queue_name).queue_url

  puts "Begin receipt of any messages using receive_message..."
  receive_message_result = sqs.receive_message({
    queue_url: receive_queue_url,
    attribute_names: ["All"], # Receive all available built-in message attributes.
    message_attribute_names: ["All"], # Receive any custom message attributes.
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Received #{receive_message_result.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using receive_message for a queue named
 '#{receive_queue_name}', as it does not exist."
end

# 2. Using Aws::SQS::QueuePoller.
begin
  puts "Begin receipt of any messages using Aws::SQS::QueuePoller..."
  puts "(Will keep polling until no more messages available for at least 60 seconds.)"
  poller = Aws::SQS::QueuePoller.new(receive_queue_url)

  poller_stats = poller.poll({
    max_number_of_messages: 10,
    idle_timeout: 60 # Stop polling after 60 seconds of no more messages available
 (polls indefinitely by default).
  }) do |messages|
    messages.each do |message|
      puts "Message body: #{message.body}"
    end
  end
  # Note: If poller.poll is successful, all received messages are automatically deleted
 from the queue.

  puts "Poller stats:"
```

```
  puts "  Polling started at: #{poller_stats.polling_started_at}"
  puts "  Polling stopped at: #{poller_stats.polling_stopped_at}"
  puts "  Last message received at: #{poller_stats.last_message_received_at}"
  puts "  Number of polling requests: #{poller_stats.request_count}"
  puts "  Number of received messages: #{poller_stats.received_message_count}"
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using Aws::SQS::QueuePoller for a queue named
 '#{receive_queue_name}', as it does not exist."
end
```

Ricezione di messaggi utilizzando la QueuePoller classe in Amazon SQS

L'esempio seguente utilizza la classe di `QueuePoller` utilità per visualizzare il corpo di tutti i messaggi nella coda Amazon SQS con l'URL URL nella `us-west-2` regione ed elimina il messaggio. Dopo circa 15 secondi di inattività, lo script scade.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(idle_timeout: 15) do |msg|
  puts msg.body
end
```

L'esempio seguente esegue un loop nella coda di Amazon SQS con l'*URL* URL e attende fino a una durata di secondi.

Puoi ottenere l'URL corretto eseguendo l'esempio di Amazon SQS in Getting Information about All Queues in Amazon SQS.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(wait_time_seconds: duration, idle_timeout: duration + 1) do |msg|
  puts msg.body
end
```

L'esempio seguente esegue un loop nella coda di Amazon SQS con l'URLURL e fornisce i secondi di timeout di visibilità necessari per elaborare il messaggio, rappresentato dal metodo. `do_something`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Process the message
```

```
def do_something(msg)
  puts msg.body
end

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(visibility_timeout: timeout, idle_timeout: timeout + 1) do |msg|
  do_something(msg)
end
```

L'esempio seguente esegue un loop nella coda di Amazon SQS con URL l'URL e modifica i secondi di timeout di visibilità per qualsiasi messaggio che richieda un'ulteriore elaborazione con il metodo. `do_something2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Process the message
def do_something(_)
  true
end

# Do additional processing
def do_something2(msg)
  puts msg.body
end

Aws.config.update({region: 'us-west-2'})

poller = Aws::SQS::QueuePoller.new(URL)
```

```
poller.poll(idle_timeout: timeout + 1) do |msg|
  if do_something(msg)
    # need more time for processing
    poller.change_message_visibility_timeout(msg, timeout)

    do_something2(msg)
  end
end
```

Reindirizzamento di lettere morte in Amazon SQS

L'esempio seguente reindirizza tutte le lettere morte dalla coda con l'URL URL alla coda con l'ARN. ARN

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.set_queue_attributes({
  queue_url: URL,
  attributes:
    {
      'RedrivePolicy' => "{\"maxReceiveCount\":\"5\", \"deadLetterTargetArn\":
\"#{ARN}\"}"
    }
})
```

Eliminazione di una coda in Amazon SQS

L'esempio seguente elimina la coda Amazon SQS con l'URLURL nella regione. `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

Abilitazione della pubblicazione di una risorsa in una coda in Amazon SQS

L'esempio seguente consente alla risorsa con l'ARN di `my-resource-arn` pubblicare nella coda con l'ARN `my-queue-arn` e l'URL nella regione. `my-queue-url` `us-west-2`

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-west-2')

policy  = '{
  "Version":"2008-10-17",
  "Id":' + my-queue-arn + '/SQSDefaultPolicy",
  "Statement":[{
```

```
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
    "Principal":{
      "AWS":"*"
    },
    "Action":["SQS:SendMessage"],
    "Resource":"' + my-queue-arn + '",
    "Condition":{
      "ArnEquals":{
        "AWS:SourceArn":"' + my-resource-arn + '"}
     }
  }]
}'

sqs.set_queue_attributes({
  queue_url: my-queue-url,
  attributes: {
    Policy: policy
  }
})
```

Utilizzo di una coda di lettere morte in Amazon SQS

Amazon SQS fornisce supporto per le code di lettere morte. Una coda di lettere morte è una coda che altre code (di origine) possono indirizzare per i messaggi che non possono essere elaborati correttamente. È possibile mettere da parte e isolare questi messaggi nella coda delle lettere morte per determinare il motivo per cui la loro elaborazione non è riuscita. Per ulteriori informazioni sulle code di lettere morte, consulta Using Amazon SQS Dead Letter Queues.

In questo esempio, utilizzi l'AWSSDK for Ruby con Amazon SQS per:

1. Crea una coda che rappresenti una coda di lettere morte utilizzando Aws: :SQS: :Client #create_queue.
2. Associa la coda di lettere morte a una coda esistente utilizzando Aws: :SQS: :Client #set_queue_attributes.
3. Invia un messaggio alla coda esistente utilizzando Aws: :SQS: :Client #send_message.
4. Effettua il polling della coda utilizzando Aws: :SQS::. QueuePoller
5. Ricevi messaggi nella coda delle lettere morte utilizzando Aws: :SQS: :Client #receive_message.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

È inoltre necessario utilizzare my-queue AWS Management Console per creare la coda esistente.

> **Note**
>
> Per motivi di semplicità, questo codice di esempio non dimostra Aws: :SQS: :Client #add_permission. In uno scenario reale, dovresti sempre limitare l'accesso ad azioni come SendMessage,, e. ReceiveMessage DeleteMessage DeleteQueue In caso contrario, potrebbero verificarsi la divulgazione di informazioni, l'interruzione del servizio o l'inserimento di messaggi nelle code.

Esempio

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to:
# 1. Create a queue representing a dead letter queue.
# 2. Associate the dead letter queue with an existing queue.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

# Uncomment for Windows.
# Aws.use_bundled_cert!
```

```
sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue representing a dead letter queue.
dead_letter_queue_name = "dead-letter-queue"

sqs.create_queue({
  queue_name: dead_letter_queue_name
})

# Get the dead letter queue's URL and ARN, so that you can associate it with an
 existing queue.
dead_letter_queue_url = sqs.get_queue_url(queue_name: dead_letter_queue_name).queue_url

dead_letter_queue_arn = sqs.get_queue_attributes({
  queue_url: dead_letter_queue_url,
  attribute_names: ["QueueArn"]
}).attributes["QueueArn"]

# Associate the dead letter queue with an existing queue.
begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Use a redrive policy to specify the dead letter queue and its behavior.
  redrive_policy = {
    "maxReceiveCount" => "5", # After the queue receives the same message 5 times, send
 that message to the dead letter queue.
    "deadLetterTargetArn" => dead_letter_queue_arn
  }.to_json

  sqs.set_queue_attributes({
    queue_url: queue_url,
    attributes: {
      "RedrivePolicy" => redrive_policy
    }
  })

rescue Aws::SQS::Errors::NonExistentQueue
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
end

# Send a message to the queue.
```

```
puts "Sending a message..."

sqs.send_message({
  queue_url: queue_url,
  message_body: "I hope I get moved to the dead letter queue."
})

30.downto(0) do |i|
  print "\rWaiting #{i} second(s) for sent message to be receivable..."
  sleep(1)
end

puts "\n"

poller = Aws::SQS::QueuePoller.new(queue_url)
# Receive 5 messages max and stop polling after 20 seconds of no received messages.
poller.poll(max_number_of_messages:5, idle_timeout: 20) do |messages|
  messages.each do |msg|
    puts "Received message ID: #{msg.message_id}"
  end
end

# Check to see if Amazon SQS moved the message to the dead letter queue.
receive_message_result = sqs.receive_message({
  queue_url: dead_letter_queue_url,
  max_number_of_messages: 1
})

if receive_message_result.messages.count > 0
  puts "\n#{receive_message_result.messages[0].body}"
else
  puts "\nNo messages received."
end
```

Specificazione del timeout di visibilità dei messaggi in Amazon SQS

In Amazon SQS, subito dopo la ricezione di un messaggio, questo rimane in coda. Per impedire ad altri consumatori di elaborare nuovamente il messaggio, Amazon SQS imposta un timeout di visibilità. Questo è un periodo di tempo durante il quale Amazon SQS impedisce ad altri componenti che consumano di ricevere ed elaborare il messaggio. Per ulteriori informazioni, consulta Timeout visibilità.

In questo esempio, utilizzi l'AWSSDK for Ruby con Amazon SQS per:

1. Ottieni l'URL di una coda esistente utilizzando Aws: :SQS: :Client #get_queue_url.
2. Ricevi fino a 10 messaggi utilizzando Aws: :SQS: :Client #receive_message.
3. Specificate l'intervallo di tempo durante il quale i messaggi non sono visibili dopo la loro ricezione, utilizzando Aws: :SQS: :Client #change_message_visibility.

Prerequisiti

Prima di eseguire il codice di esempio, è necessario installare e configurare l'AWSSDK for Ruby, come descritto in:

- Installazione dell'AWSSDK per Ruby
- Configurazione dell'AWSSDK per Ruby

È inoltre necessario creare la coda my-queue, operazione che è possibile eseguire nella console Amazon SQS.

Esempio

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# Demonstrates how to specify the time interval during which messages to a queue are
 not visible after being received.

require 'aws-sdk-sqs'  # v2: require 'aws-sdk'

sqs = Aws::SQS::Client.new(region: 'us-east-1')

begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
```

```
  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not be visible for 30 seconds after
 first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."

rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{receive_queue_name}', as it does
 not exist."
end
```

Esempi di Amazon WorkDocs

Puoi utilizzare i seguenti esempi per accedere ad Amazon WorkDocs (Amazon WorkDocs) utilizzando l'AWSSDK for Ruby. Per ulteriori informazioni su Amazon WorkDocs, consulta la documentazione di Amazon WorkDocs .

Hai bisogno dell'ID della tua organizzazione per utilizzare questi esempi. Ottieni l'ID dell'organizzazione dalla AWS console utilizzando i seguenti passaggi:

- Seleziona il AWS Directory Service
- Selezionare `Directories`

L'ID dell'organizzazione è quello `Directory ID` corrispondente al tuo WorkDocs sito Amazon.

Examples (Esempi)

Argomenti

- Elencazione di utenti
- Elenco dei documenti utente

Elencazione di utenti

L'esempio seguente elenca i nomi, gli indirizzi e-mail e le cartelle principali di tutti gli utenti dell'organizzazione. Scegliete di `Copy` salvare il codice localmente oppure consultate il collegamento all'esempio completo alla fine di questo argomento.

1. Richiedi il modulo AWS SDK for Ruby e crea un WorkDocs client Amazon.

2. Chiama `describe_users` con l'ID della tua organizzazione e ottieni tutti i nomi utente in ordine crescente.

1. Visualizza le informazioni sugli utenti.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-workdocs'  # v2: require 'aws-sdk'
```

```
client = Aws::WorkDocs::Client.new(region: 'us-west-2')

# Set to the OrganizationId of your WorkDocs site
orgId = 'd-123456789c'

resp = client.describe_users({
  organization_id: orgId,
  include: "ALL", # accepts ALL, ACTIVE_PENDING
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
  sort: "USER_NAME", # accepts USER_NAME, FULL_NAME, STORAGE_LIMIT, USER_STATUS,
 STORAGE_USED
})

resp.users.each do |user|
  puts "First name:  #{user.given_name}"
  puts "Last name:   #{user.surname}"
  puts "Email:       #{user.email_address}"
  puts "Root folder: #{user.root_folder_id}"
  puts
end
```

Vedi l'esempio completo su GitHub.

Elenco dei documenti utente

L'esempio seguente elenca i documenti per un utente. Scegliete di Copy salvare il codice localmente oppure consultate il collegamento all'esempio completo alla fine di questo argomento.

1. Richiede il AWS modulo SDK for Ruby.
2. Crea un metodo di supporto per ottenere la cartella principale di un utente.
3. Crea un WorkDocs client Amazon.
4. Ottieni la cartella principale di quell'utente.

5. Chiama `describe_folder_contents` per visualizzare il contenuto della cartella in ordine crescente.

6. Visualizza il nome, la dimensione (in byte), la data dell'ultima modifica, l'ID del documento e l'ID della versione per ogni documento nella cartella principale dell'utente.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
```

```
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

require 'aws-sdk-workdocs'  # v2: require 'aws-sdk'

def get_user_folder(client, orgId, user_email)
  root_folder = ''

  resp = client.describe_users({
    organization_id: orgId,
  })

  # resp.users should have only one entry
  resp.users.each do |user|
    if user.email_address == user_email
      root_folder = user.root_folder_id
    end
  end

  return root_folder
end

client = Aws::WorkDocs::Client.new(region: 'us-west-2')

# Set to the email address of a user
user_email = 'someone@somewhere'

# Set to the OrganizationId of your WorkDocs site.
orgId = 'd-123456789c'

user_folder = get_user_folder(client, orgId, user_email)

if user_folder == ''
  puts 'Could not get root folder for user with email address ' + user_email
  exit(1)
end
```

```
resp = client.describe_folder_contents({
  folder_id: user_folder, # required
  sort: "NAME", # accepts DATE, NAME
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
})

resp.documents.each do |doc|
  md = doc.latest_version_metadata

  puts "Name:          #{md.name}"
  puts "Size (bytes):  #{md.size}"
  puts "Last modified: #{doc.modified_timestamp}"
  puts "Doc ID:        #{doc.id}"
  puts "Version ID:    #{md.id}"
  puts
end
```

Vedi l'esempio completo su. GitHub

Esempi di codice SDK for Ruby

Gli esempi di codice riportati in questo argomento mostrano come utilizzare AWS SDK for Ruby with AWS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Esempi cross-service: applicazioni di esempio che funzionano su più servizi Servizi AWS.

Esempi

- Azioni e scenari che utilizzano SDK for Ruby
- Esempi interservizi che utilizzano SDK for Ruby

Azioni e scenari che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando with. AWS SDK for Ruby Servizi AWS

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Servizi

- CloudTrail esempi che utilizzano SDK for Ruby
- CloudWatch esempi che utilizzano SDK for Ruby
- Esempi di DynamoDB con SDK for Ruby
- Esempi di Amazon EC2 che utilizzano SDK for Ruby

- Esempi di Elastic Beanstalk con SDK for Ruby
- EventBridge esempi che utilizzano SDK for Ruby
- AWS Glue esempi che utilizzano SDK for Ruby
- Esempi IAM che utilizzano SDK for Ruby
- Esempi di Kinesis con SDK for Ruby
- AWS KMS esempi che utilizzano SDK for Ruby
- Esempi di Lambda con SDK for Ruby
- Esempi di Amazon Polly con SDK for Ruby
- Esempi di Amazon RDS con SDK for Ruby
- Esempi di Amazon S3 che utilizzano SDK for Ruby
- Esempi di Amazon SES con SDK for Ruby
- Esempi di API Amazon SES v2 con SDK for Ruby
- Esempi di Amazon SNS che utilizzano SDK for Ruby
- Esempi di Amazon SQS con SDK for Ruby
- AWS STS esempi che utilizzano SDK for Ruby
- WorkDocs Esempi di Amazon che utilizzano SDK for Ruby

CloudTrail esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with CloudTrail.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Crea percorsi

Il seguente esempio di codice mostra come creare un AWS CloudTrail trail.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta la CreateTrailsezione AWS SDK for Ruby API Reference.

Elimina percorso

Il seguente esempio di codice mostra come eliminare un AWS CloudTrail trail.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

- Per i dettagli sull'API, consulta la DeleteTrailsezione AWS SDK for Ruby API Reference.

Elenca gli eventi del percorso

Il seguente esempio di codice mostra come elencare gli eventi del AWS CloudTrail trail.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta la LookupEventssezione AWS SDK for Ruby API Reference.

Elenca i percorsi

Il seguente esempio di codice mostra come elencare i AWS CloudTrail percorsi.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta la ListTrailssezione AWS SDK for Ruby API Reference.

CloudWatch esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with CloudWatch.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Creazione un allarme per un parametro

Il seguente esempio di codice mostra come creare o aggiornare un CloudWatch allarme Amazon e associarlo alla metrica, all'espressione matematica della metrica, al modello di rilevamento delle anomalie o alla query Metrics Insights specificati.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
```

```
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
```

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

- Per i dettagli sull'API, consulta la PutMetricAlarmsezione AWS SDK for Ruby API Reference.

Descrizione di allarmi

Il seguente esempio di codice mostra come descrivere gli CloudWatch allarmi Amazon.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta la DescribeAlarmssezione AWS SDK for Ruby API Reference.

Descrivi allarmi per un parametro

Il seguente esempio di codice mostra come descrivere gli CloudWatch allarmi Amazon per una metrica.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts "-" * 16
      puts "Name:           " + alarm.alarm_name
      puts "State value:    " + alarm.state_value
      puts "State reason:   " + alarm.state_reason
      puts "Metric:         " + alarm.metric_name
      puts "Namespace:      " + alarm.namespace
      puts "Statistic:      " + alarm.statistic
      puts "Period:         " + alarm.period.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      puts "Comp. operator: " + alarm.comparison_operator

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

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

```
          puts "  " + a
        end
      end

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

      puts "Dimensions:"
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: " + d.name + ", Value: " + d.value
        end
      else
        puts "  None for this alarm."
      end
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la DescribeAlarmsForMetricsezione AWS SDK for Ruby API Reference.

Disattivare le operazioni di allarme

Il seguente esempio di codice mostra come disabilitare le azioni di CloudWatch allarme di Amazon.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
```

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

# Example usage:
def run_me
  alarm_name = "ObjectsInBucket"
  alarm_description = "Objects exist in this bucket for more than 1 day."
  metric_name = "NumberOfObjects"
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ["arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic"]
  namespace = "AWS/S3"
  statistic = "Average"
  dimensions = [
    {
      name: "BucketName",
      value: "doc-example-bucket"
    },
    {
      name: "StorageType",
      value: "AllStorageTypes"
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = "Count"
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = "GreaterThanThreshold" # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

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

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
```

```
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la DisableAlarmActionssezione AWS SDK for Ruby API Reference.

Elencare parametri

Il seguente esempio di codice mostra come elencare i metadati per le CloudWatch metriche di Amazon. Per ottenere i dati per una metrica, usa le GetMetricData azioni o. GetMetricStatistics

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
```

```
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "    Dimensions:"
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

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

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
```

```
    "example.com",
    5_885.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la ListMetricssezione AWS SDK for Ruby API Reference.

Inserimento dei dati in un parametro

Il seguente esempio di codice mostra come pubblicare punti dati metrici su Amazon CloudWatch.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-cloudwatch"

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
```

```
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- Per i dettagli sull'API, consulta la PutMetricDatasezione AWS SDK for Ruby API Reference.

Esempi di DynamoDB con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con DynamoDB.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

Azioni

Creare una tabella

Il seguente esempio di codice mostra come creare una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
```

```
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        {attribute_name: "year", key_type: "HASH"},  # Partition key
        {attribute_name: "title", key_type: "RANGE"}  # Sort key
      ],
      attribute_definitions: [
        {attribute_name: "year", attribute_type: "N"},
        {attribute_name: "title", attribute_type: "S"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la CreateTablesezione AWS SDK for Ruby API Reference.

Eliminazione di una tabella

Il seguente esempio di codice mostra come eliminare una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK for Ruby API Reference.

Elimina una voce da una tabella

Il seguente esempio di codice mostra come eliminare un elemento da una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

```
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: {"year" => year, "title" => title})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la DeleteItemsezione AWS SDK for Ruby API Reference.

Ottieni un elemento da una tabella

Il seguente esempio di codice mostra come ottenere un elemento da una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Gets movie data from the table for a specific movie.
  #
```

```
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: {"year" => year, "title" => title})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la GetItemsezione AWS SDK for Ruby API Reference.

Ottieni informazioni su una tabella

Il seguente esempio di codice mostra come ottenere informazioni su una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

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

- Per i dettagli sull'API, consulta la DescribeTablesezione AWS SDK for Ruby API Reference.

Elencare tabelle

Il seguente esempio di codice mostra come elencare le tabelle DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Determina se esiste una tabella.

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

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

```
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

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

- Per i dettagli sull'API, consulta la ListTablessezione AWS SDK for Ruby API Reference.

Inserisci un elemento in una tabella

Il seguente esempio di codice mostra come inserire un elemento in una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

- Per i dettagli sull'API, consulta la PutItemsezione AWS SDK for Ruby API Reference.

Esecuzione di una query su una tabella

Il seguente esempio di codice mostra come eseguire una query su una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

- Per ulteriori informazioni sulle API, consulta Query nella Documentazione di riferimento delle API AWS SDK for Ruby .

Esecuzione di un'istruzione PartiQL

Il seguente esempio di codice mostra come eseguire un'istruzione PartiQL su una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Selezione di un elemento mediante PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Aggiornamento di un elemento mediante PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

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

```
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Aggiunta di un elemento mediante PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

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

Eliminazione di un elemento mediante PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- Per i dettagli sull'API, consulta la ExecuteStatementsezione AWS SDK for Ruby API Reference.

Esecuzione di batch di istruzioni PartiQL

Il seguente esempio di codice mostra come eseguire batch di istruzioni PartiQL su una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Lettura di un batch di elementi mediante PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table

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

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

Eliminazione di un batch di elementi mediante PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table

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

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
```

```
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({statements: request_items})
  end
```

- Per i dettagli sull'API, consulta la BatchExecuteStatementsezione AWS SDK for Ruby API Reference.

Esegui la scansione di una tabella

Il seguente esempio di codice mostra come eseguire la scansione di una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
```

```
    movies = []
    scan_hash = {
      filter_expression: "#yr between :start_yr and :end_yr",
      projection_expression: "#yr, title, info.rating",
      expression_attribute_names: {"#yr" => "year"},
      expression_attribute_values: {
        ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Per informazioni dettagliate sulle API, consulta Scan nella Documentazione di riferimento per le API AWS SDK for Ruby .

Aggiorna un elemento in una tabella

Il seguente esempio di codice mostra come aggiornare un elemento in una tabella DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class DynamoDBBasics
```

```
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)

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

- Per i dettagli sull'API, consulta la UpdateItemsezione AWS SDK for Ruby API Reference.

Scrittura di un batch di elementi

Il seguente esempio di codice mostra come scrivere un batch di elementi DynamoDB.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must contain
 at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
      movies[index, slice_size].each do |movie|
        movie_items.append({put_request: { item: movie }})
      end
      @dynamo_resource.client.batch_write_item({request_items: { @table.name =>
 movie_items }})
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la BatchWriteItemsezione AWS SDK for Ruby API Reference.

Scenari

Nozioni di base sull'utilizzo di tabelle, elementi e query

L'esempio di codice seguente mostra come:

- Crea una tabella in grado di contenere i dati del filmato.
- Inserisci, ottieni e aggiorna un singolo filmato nella tabella.
- Scrivi i dati del filmato nella tabella da un file JSON di esempio.
- Esegui una query sui filmati che sono stati rilasciati in un dato anno.
- Cerca i filmati che sono stati distribuiti in diversi anni.
- Elimina un filmato dalla tabella, quindi elimina la tabella.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe che incapsula una tabella DynamoDB.

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
  @table = @dynamo_resource.create_table(
    table_name: table_name,
    key_schema: [
      {attribute_name: "year", key_type: "HASH"},  # Partition key
      {attribute_name: "title", key_type: "RANGE"}  # Sort key
    ],
    attribute_definitions: [
      {attribute_name: "year", attribute_type: "N"},
```

```
        {attribute_name: "title", attribute_type: "S"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Crea una funzione helper per scaricare ed estrarre il file JSON di esempio.

```
  # 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
```

Esegui uno scenario interattivo per creare la tabella ed eseguire azioni su di essa.

```
 table_name = "doc-example-table-movies-#{rand(10**4)}"
 scaffold = Scaffold.new(table_name)
 dynamodb_wrapper = DynamoDBBasics.new(table_name)

 new_step(1, "Create a new DynamoDB table if none already exists.")
 unless scaffold.exists?(table_name)
   puts("\nNo such table: #{table_name}. Creating it...")
   scaffold.create_table(table_name)
   print "Done!\n".green
 end

 new_step(2, "Add a new record to the DynamoDB table.")
 my_movie = {}
 my_movie[:title] = CLI::UI::Prompt.ask("Enter the title of a movie to add to the
table. E.g. The Matrix")
 my_movie[:year] = CLI::UI::Prompt.ask("What year was it released? E.g. 1989").to_i
 my_movie[:rating] = CLI::UI::Prompt.ask("On a scale of 1 - 10, how do you rate it?
E.g. 7").to_i
 my_movie[:plot] = CLI::UI::Prompt.ask("Enter a brief summary of the plot. E.g. A
man awakens to a new reality.")
 dynamodb_wrapper.add_item(my_movie)
 puts("\nNew record added:")
 puts JSON.pretty_generate(my_movie).green
 print "Done!\n".green

 new_step(3, "Update a record in the DynamoDB table.")
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, "Get a record from the DynamoDB table.")
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(5, "Write a batch of items into the DynamoDB table.")
 download_file = "moviedata.json"
 puts("Downloading movie database to #{download_file}...")
```

```
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
 print "Done!\n".green

 new_step(5, "Query for a batch of items by key.")
 loop do
   release_year = CLI::UI::Prompt.ask("Enter a year between 1972 and 2018, e.g.
1999:").to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie["title"]}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
     break if !continue.eql?("y")
   end
 end
 print "\nDone!\n".green

 new_step(6, "Scan for a batch of items using a filter expression.")
 years = {}
 years[:start] = CLI::UI::Prompt.ask("Enter a starting year between 1972 and
2018:")
 years[:end] = CLI::UI::Prompt.ask("Enter an ending year between 1972 and 2018:")
 releases = dynamodb_wrapper.scan_items(years)
 if !releases.empty?
   puts("Found #{releases.length} movies.")
   count = Question.ask(
     "How many do you want to see? ", method(:is_int), in_range(1,
releases.length))
   puts("Here are your #{count} movies:")
   releases.take(count).each do |release|
     puts("\t#{release["title"]}")
   end
 else
   puts("I don't know about any movies released between #{years[:start]} "\
        "and #{years[:end]}.")
 end
```

```
  print "\nDone!\n".green

  new_step(7, "Delete an item from the DynamoDB table.")
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
")
  if answer.eql?("y")
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, "Delete the DynamoDB table.")
  answer = CLI::UI::Prompt.ask("Delete the table? (y/n)")
  if answer.eql?("y")
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo.")
rescue Errno::ENOENT
  true
end
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Esecuzione di una query su una tabella mediante batch di istruzioni PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un batch di elementi mediante più istruzioni SELECT.
- Aggiunta di un batch di articoli eseguendo più istruzioni INSERT.
- Aggiornamento di un batch di elementi mediante più istruzioni UPDATE.
- Eliminazione di un batch di elementi mediante più istruzioni DELETE.

SDK per Ruby

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esecuzione di uno scenario che crea una tabella ed esegue query PartiQL in batch.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLBatch.new(table_name)

new_step(1, "Create a new DynamoDB table if none already exists.")
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, "Populate DynamoDB table with movie data.")
download_file = "moviedata.json"
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, "Select a batch of items from the movies table.")
puts "Let's select some popular movies for side-by-side comparison."
```

```
  response = sdk.batch_execute_select([["Mean Girls", 2004], ["Goodfellas", 1977],
 ["The Prancing of the Lambs", 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, "Delete a batch of items from the movies table.")
  sdk.batch_execute_write([["Mean Girls", 2004], ["Goodfellas", 1977], ["The
 Prancing of the Lambs", 2005]])
  print "\nDone!\n".green

  new_step(5, "Delete the table.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
  end
end
```

- Per i dettagli sull'API, consulta la BatchExecuteStatementsezione AWS SDK for Ruby API Reference.

Esecuzione di una query mediante PartiQL

L'esempio di codice seguente mostra come:

- Ricezione di un articolo eseguendo un'istruzione SELECT.
- Aggiunta di un elemento eseguendo un'istruzione INSERT.
- Aggiornamento di un elemento eseguendo un'istruzione UPDATE.
- Eliminazione di un elemento eseguendo un'istruzione DELETE.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esecuzione di uno scenario che crea una tabella ed esegue query PartiQL.

```
  table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
```

```
  scaffold = Scaffold.new(table_name)
  sdk = DynamoDBPartiQLSingle.new(table_name)

  new_step(1, "Create a new DynamoDB table if none already exists.")
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, "Populate DynamoDB table with movie data.")
  download_file = "moviedata.json"
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, "Select a single item from the movies table.")
  response = sdk.select_item_by_title("Star Wars")
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print "#{response.items.first}".yellow
  print "\n\nDone!\n".green

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

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

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

  new_step(7, "Delete the table.")
  if scaffold.exists?(table_name)
    scaffold.delete_table
```

```
  end
end
```

- Per i dettagli sull'API, consulta la ExecuteStatementsezione AWS SDK for Ruby API Reference.

Esempi di Amazon EC2 che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon EC2. AWS SDK for Ruby

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Allocare un indirizzo IP elastico

Il seguente esempio di codice mostra come allocare un indirizzo IP elastico per Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
```

```
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: "vpc")
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return "Error"
end
```

- Per i dettagli sull'API, consulta la AllocateAddresssezione AWS SDK for Ruby API Reference.

Associazione di un indirizzo IP elastico a un'istanza

Il seguente esempio di codice mostra come associare un indirizzo IP elastico a un'istanza Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
```

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

- Per i dettagli sull'API, consulta la AssociateAddresssezione AWS SDK for Ruby API Reference.

Creazione di un Amazon Virtual Private Cloud (Amazon VPC)

L'esempio di codice seguente mostra come creare un Amazon Virtual Private Cloud (Amazon VPC).

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
```

```
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return false
end

# Example usage:
def run_me
  cidr_block = ""
  tag_key = ""
  tag_value = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateVpcconsulta AWS SDK for Ruby API Reference.

Creazione di una tabella di routing

L'esempio di codice seguente mostra come creare una tabella di routing e associarla alla sottorete Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
```

```
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Added tags to route table."
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts "Created route with destination CIDR block " \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  return true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts "If the route table was created but not associated, you should " \
    "clean up by deleting the route table."
  return false
end

# Example usage:
def run_me
  vpc_id = ""
  subnet_id = ""
  gateway_id = ""
  destination_cidr_block = ""
  tag_key = ""
```

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

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

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
```

```
    puts "Route table created and associated."
  else
    puts "Route table not created or not associated."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateRouteTableconsulta AWS SDK for Ruby API Reference.

Creazione di un gruppo di sicurezza

Il seguente esempio di codice mostra come creare un gruppo di sicurezza Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require "aws-sdk-ec2"

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
```

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

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

```
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client,
  security_group_id,
  ip_protocol,
  from_port,
  to_port,
  cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

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

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

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

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

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

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

  print "\n"
end

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

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

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      puts "-" * (sg.group_name.length + 13)
      puts "Name:        #{sg.group_name}"
      puts "Description: #{sg.description}"
      puts "Group ID:    #{sg.group_id}"
      puts "Owner ID:    #{sg.owner_id}"
      puts "VPC ID:      #{sg.vpc_id}"

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

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

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

```
# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Example usage:
def run_me
  group_name = ""
  description = ""
  vpc_id = ""
  ip_protocol_http = ""
  from_port_http = ""
  to_port_http = ""
  cidr_ip_range_http = ""
  ip_protocol_ssh = ""
  from_port_ssh = ""
  to_port_ssh = ""
  cidr_ip_range_ssh = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-security-group.rb " \
      "GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 " \
      "CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 " \
```

```
    "CIDR_IP_RANGE_2 REGION"
  puts "Example: ruby ec2-ruby-example-security-group.rb " \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  group_name = "my-security-group"
  description = "This is my security group."
  vpc_id = "vpc-6713dfEX"
  ip_protocol_http = "tcp"
  from_port_http = "80"
  to_port_http = "80"
  cidr_ip_range_http = "0.0.0.0/0"
  ip_protocol_ssh = "tcp"
  from_port_ssh = "22"
  to_port_ssh = "22"
  cidr_ip_range_ssh = "0.0.0.0/0"
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

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

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

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

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

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

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

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

```
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateSecurityGroupconsulta AWS SDK for Ruby API Reference.

Creazione di una coppia di chiavi di sicurezza

Il seguente esempio di codice mostra come creare una coppia di chiavi di sicurezza per Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
```

```
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + ".pem")
  File.open(filename, "w") { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    "already exists."
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  return false
end

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

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
```

```
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return false
end

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateKeyPairconsulta AWS SDK for Ruby API Reference.

Creazione di una sottorete

Gli esempi di codice seguenti mostrano come creare una sottorete Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
```

```
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return false
end

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

```
  elsif ARGV.count.zero?
    vpc_id = "vpc-6713dfEX"
    cidr_block = "10.0.0.0/24"
    availability_zone = "us-west-2a"
    tag_key = "my-key"
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateSubnetconsulta AWS SDK for Ruby API Reference.

Descrivere le regioni

Il seguente esempio di codice mostra come descrivere le regioni Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Endpoint\n"
  print "-" * max_region_string_length
  print "  "
  print "-" * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print " " * (max_region_string_length - region.region_name.length)
    print "  "
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
```

```
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Zone"
  print " " * (max_zone_string_length - "Zone".length)
  print "  State\n"
  print "-" * max_region_string_length
  print "  "
  print "-" * max_zone_string_length
  print "  "
  print "-" * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print " " * (max_region_string_length - zone.region_name.length)
    print "  "
    print zone.zone_name
    print " " * (max_zone_string_length - zone.zone_name.length)
    print "  "
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts "  Messages for this zone:"
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ""
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, DescribeRegionsconsulta AWS SDK for Ruby API Reference.

Descrivere le istanze

Il seguente esempio di codice mostra come descrivere le istanze Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts "No instances found."
  else
    puts "Instances -- ID, state:"
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts "Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = "us-west-2"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, DescribeInstancesconsulta AWS SDK for Ruby API Reference.

Rilascio di un indirizzo IP elastico

Il seguente esempio di codice mostra come rilasciare un indirizzo IP elastico.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end
```

- Per i dettagli sull'API, ReleaseAddressconsulta AWS SDK for Ruby API Reference.

Avviare un'istanza

Il seguente esempio di codice mostra come avviare un'istanza Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

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

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when "pending"
      puts "Error starting instance: the instance is pending. Try again later."
      return false
    when "running"
      puts "The instance is already running."
      return true
    when "terminated"
      puts "Error starting instance: " \
```

```
        "the instance is terminated, so you cannot start it."
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance started."
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return false
end

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

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

  puts "Attempting to start instance '#{instance_id}' " \
    "(this might take a few minutes)..."
  unless instance_started?(ec2_client, instance_id)
    puts "Could not start instance."
  end
```

```
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, StartInstancesconsulta AWS SDK for Ruby API Reference.

Arrestare un'istanza

Il seguente esempio di codice mostra come interrompere un'istanza Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

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

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when "stopping"
      puts "The instance is already stopping."
```

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

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

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

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

```
  puts "Attempting to stop instance '#{instance_id}' " \
    "(this might take a few minutes)..."
  unless instance_stopped?(ec2_client, instance_id)
    puts "Could not stop instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, StopInstancesconsulta AWS SDK for Ruby API Reference.

Terminare un'istanza

Il seguente esempio di codice mostra come terminare un'istanza Amazon EC2.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ec2"

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

```
  if response.instance_statuses.count.positive? &&
    response.instance_statuses[0].instance_state.name == "terminated"

    puts "The instance is already terminated."
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts "Instance terminated."
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return false
end

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

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

  puts "Attempting to terminate instance '#{instance_id}' " \
    "(this might take a few minutes)..."
```

```
  unless instance_terminated?(ec2_client, instance_id)
    puts "Could not terminate instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, TerminateInstancesconsulta AWS SDK for Ruby API Reference.

Esempi di Elastic Beanstalk con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con Elastic Beanstalk.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Descrivi app

Il seguente esempio di codice mostra come descrivere un' AWS Elastic Beanstalk app.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DescribeApplicationsconsulta AWS SDK for Ruby API Reference.

Elenca pile

Il seguente esempio di codice mostra come elencare le AWS Elastic Beanstalk pile.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

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

  private
```

```
  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Per i dettagli sull'API, ListAvailableSolutionStacksconsulta AWS SDK for Ruby API Reference.

Aggiorna l'app

Il seguente esempio di codice mostra come aggiornare un' AWS Elastic Beanstalk app.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end
```

```
  private

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

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

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

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

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, ".zip")
    @eb_client.create_application_version(
      process: false,
```

```
      application_name: @app_name,
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

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

- Per i dettagli sull'API, UpdateApplicationconsulta AWS SDK for Ruby API Reference.

EventBridge esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with EventBridge.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Scenari

Creazione e attivazione di una regola

Il seguente esempio di codice mostra come creare e attivare una regola in Amazon EventBridge.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Chiama le funzioni nell'ordine corretto.

```
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"
```

Verifica se l'argomento Amazon Simple Notification Service (Amazon SNS) esiste tra quelli forniti a questa funzione.

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

def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

Verifica se l'argomento specificato esiste tra quelli disponibili per il chiamante in 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? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

Crea un argomento in Amazon SNS e quindi registrati con un indirizzo e-mail per ricevere notifiche su quell'argomento.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
```

```
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Verifica se il ruolo specificato AWS Identity and Access Management (IAM) esiste tra quelli forniti a questa funzione.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

Verifica se il ruolo specificato esiste tra quelli disponibili per il chiamante in IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

Crea un ruolo in 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."
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts "If the role was created, you must add the access policy " \
    "to the role yourself, or delete the role yourself and try again."
  return "Error"
end
```

Verifica se la EventBridge regola specificata esiste tra quelle fornite a questa funzione.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end
```

Verifica se la regola specificata esiste tra quelle disponibili per il chiamante in EventBridge.

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

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

Crea una regola in EventBridge.

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

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

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

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

Verifica se il gruppo di log specificato esiste tra quelli disponibili per il chiamante in Amazon CloudWatch Logs.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

Crea un gruppo di log in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
```

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

Scrivi un evento in un flusso di log in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
```

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

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

Riavvia un'istanza Amazon Elastic Compute Cloud (Amazon EC2) e aggiunge informazioni sull'attività correlata a un flusso di log in Logs. CloudWatch

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
```

```
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ""

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    "This might take a few minutes..."
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
```

```
  )

  puts "Attempting to restart the instance. This might take a few minutes..."
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance restarted."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  return true
rescue StandardError => e
  puts "Error creating log stream or stopping or restarting the instance: " \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  return false
end
```

Visualizza informazioni sull'attività per una regola in. 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
```

Visualizza le informazioni di registro per tutti i flussi di log in un gruppo di log CloudWatch Logs.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts "Attempting to display log stream data for the log group " \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: "LastEventTime",
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
```

```
      else
        puts "No log messages for this log stream."
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about the log streams or their messages: " \
    "#{e.message}"
end
```

Mostra al chiamante un promemoria affinché pulisca manualmente tutte AWS le risorse associate che non gli servono più.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
```

```
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

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

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

  # If the Amazon SNS topic doesn't exist, create it.
  topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
  unless topic_exists?(sns_client, topic_arn)
    topic_arn = create_topic(sns_client, topic_name, email_address)
    if topic_arn == "Error"
      puts "Could not create the Amazon SNS topic correctly. Program stopped."
```

```
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

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

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

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts "Could not create the Amazon CloudWatch Logs log group " \
    "correctly. Program stopped."
    manual_cleanup_notice(
```

```
        topic_name, role_name, rule_name, log_group_name, instance_id
      )
    end
  end

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - PutEvents
  - PutRule

AWS Glue esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with AWS Glue.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Scenari

Azioni

Creazione di un crawler

Il seguente esempio di codice mostra come creare un AWS Glue crawler.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateCrawlerconsulta AWS SDK for Ruby API Reference.

Creazione di una definizione di processo

Il seguente esempio di codice mostra come creare una definizione di AWS Glue processo.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateJobconsulta AWS SDK for Ruby API Reference.

Eliminazione di un crawler

Il seguente esempio di codice mostra come eliminare un AWS Glue crawler.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for Ruby API Reference.

Eliminazione di un database dal catalogo dati

Il seguente esempio di codice mostra come eliminare un database da AWS Glue Data Catalog.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for Ruby API Reference.

Eliminazione di una definizione di processo

Il seguente esempio di codice mostra come eliminare una definizione di AWS Glue processo e tutte le esecuzioni associate.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteJobconsulta AWS SDK for Ruby API Reference.

Eliminazione di una tabella da un database

Il seguente esempio di codice mostra come eliminare una tabella da un AWS Glue Data Catalog database.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteTableconsulta AWS SDK for Ruby API Reference.

Ottenimento di un crawler

Il seguente esempio di codice mostra come ottenere un AWS Glue crawler.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for Ruby API Reference.

Ottenimento di un database dal catalogo dati

Il seguente esempio di codice mostra come ottenere un database da AWS Glue Data Catalog.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for Ruby API Reference.

Ottenimento dell'esecuzione di un processo

Il seguente esempio di codice mostra come eseguire un AWS Glue job.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for Ruby API Reference.

Ottenimento di esecuzioni di un processo

Il seguente esempio di codice mostra come eseguire un AWS Glue job.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for Ruby API Reference.

Ottenimento di tabelle da un database

Il seguente esempio di codice mostra come ottenere tabelle da un database in AWS Glue Data Catalog.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetTablesconsulta AWS SDK for Ruby API Reference.

Elencazione delle definizioni di processo

Il seguente esempio di codice mostra come elencare le definizioni dei AWS Glue processi.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, ListJobsconsulta AWS SDK for Ruby API Reference.

Avvio di un crawler

Il seguente esempio di codice mostra come avviare un AWS Glue crawler.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, StartCrawlerconsulta AWS SDK for Ruby API Reference.

Avviare un'esecuzione del processo

Il seguente esempio di codice mostra come avviare l'esecuzione di un AWS Glue job.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for Ruby API Reference.

Scenari

Nozioni di base su crawler e processi

L'esempio di codice seguente mostra come:

- Crea un crawler che esegue la scansione di un bucket Amazon S3 pubblico e genera un database di metadati in formato CSV.
- Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.
- Crea un processo per estrarre i dati CSV dal bucket S3, trasformare i dati e caricare l'output in formato JSON in un altro bucket S3.
- Elenca le informazioni sulle esecuzioni dei processi, visualizza i dati trasformati e pulisci le risorse.

Per ulteriori informazioni, consulta Tutorial: Guida introduttiva a AWS Glue Studio.

SDK per Ruby

> **Note**
>
> C'è altro da sapere GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe che racchiuda le AWS Glue funzioni utilizzate nello scenario.

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

```
    @logger = logger
  end

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

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

```
  end

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

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

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

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
```

```
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

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

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

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

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
```

```
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

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

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

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

```
end
```

Creazione di una classe che esegue lo scenario.

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

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)

    new_step(1, "Create a crawler")
    puts "Checking for crawler #{crawler_name}."
    crawler = wrapper.get_crawler(crawler_name)
    if crawler == false
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}:"
      crawler = wrapper.get_crawler(crawler_name)
      puts JSON.pretty_generate(crawler).yellow
    end
    print "\nDone!\n".green

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

```
    print "\nDone!\n".green

    new_step(3, "Query the database.")
    database = wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    print "#{database}".yellow
    puts "\nThe database contains these tables:"
    tables = wrapper.get_tables(db_name)
    tables.each_with_index do |table, index|
      print "\t#{index + 1}. #{table['name']}".yellow
    end
    print "\nDone!\n".green

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

    new_step(5, "Start a new job")
    job_run_status = nil
    job_run_id = wrapper.start_job_run(
      job_name,
      db_name,
      tables[0]["name"],
      @glue_bucket.name
    )
    puts "Job #{job_name} started. Let's wait for it to run."
    until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]["job_run_state"]
      print "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
    end
    print "\nDone!\n".green

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

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

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

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

```
def main

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

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

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

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

  # Instantiate existing S3 bucket.
  s3 = Aws::S3::Resource.new(region: "us-east-1")
```

```
  s3_bucket_name = resource_names["glue_bucket"]
  s3_bucket = s3.bucket(s3_bucket_name)

  scenario = GlueCrawlerJobScenario.new(
    Aws::Glue::Client.new(region: "us-east-1"),
    iam_role,
    s3_bucket,
    @logger
  )

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

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

end
```

Create uno script ETL utilizzato da AWS Glue per estrarre, trasformare e caricare i dati durante l'esecuzione dei job.

```
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()
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Esempi IAM che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with IAM.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Collegamento di una policy a un ruolo

Il seguente esempio di codice mostra come collegare una policy IAM a un ruolo.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, allega e scollega le politiche relative ai ruoli.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
```

```
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

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

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

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

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

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

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

- Per i dettagli sulle API, consulta la sezione AWS SDK for Ruby API AttachRolePolicyReference.

Collegamento di una policy a un utente

Il seguente esempio di codice mostra come allegare una policy IAM a un utente.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, AttachUserPolicyconsulta AWS SDK for Ruby API Reference.

Creazione di una policy

Il seguente esempio di codice mostra come creare una policy IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, allega e scollega le politiche relative ai ruoli.

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

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

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

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

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

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

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

- Per i dettagli sulle API, consulta la sezione AWS SDK for Ruby API CreatePolicyReference.

Creare un ruolo

Il seguente esempio di codice mostra come creare un ruolo IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateRoleconsulta AWS SDK for Ruby API Reference.

Creazione di un ruolo collegato ai servizi

Il seguente esempio di codice mostra come creare un ruolo collegato a un servizio IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateServiceLinkedRoleconsulta AWS SDK for Ruby API Reference.

Creazione di un utente

Il seguente esempio di codice mostra come creare un utente IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateUserconsulta AWS SDK for Ruby API Reference.

Creare una chiave di accesso

Il seguente esempio di codice mostra come creare una chiave di accesso IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, disattiva ed elimina le chiavi di accesso.

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

```
  end

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

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

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

```
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

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

- Per i dettagli sull'API, consulta la sezione API CreateAccessKeyReference AWS SDK for Ruby .

Creazione di un alias per un account

Il seguente esempio di codice mostra come creare un alias per un account IAM.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, crea ed elimina gli alias degli account.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
```

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

- Per i dettagli sull'API, consulta la sezione CreateAccountAlias AWS SDK for RubyAPI Reference.

Creazione di una policy inline per un utente

L'esempio di codice seguente mostra come creare una policy IAM inline per un utente.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, PutUserPolicyconsulta AWS SDK for Ruby API Reference.

Eliminazione di un ruolo

Il seguente esempio di codice mostra come eliminare un ruolo IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

```
          end
        end
      end

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

- Per i dettagli sull'API, DeleteRoleconsulta AWS SDK for Ruby API Reference.

Eliminazione di un certificato del server

L'esempio di codice seguente mostra come eliminare un certificato del server IAM.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, aggiorna ed elimina i certificati del server.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
```

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

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

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

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

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

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

- Per i dettagli sull'API, consulta la DeleteServerCertificatesezione AWS SDK for Ruby API Reference.

Eliminazione di un ruolo collegato ai servizi

Il seguente esempio di codice mostra come eliminare un ruolo collegato a un servizio IAM.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
  # Deletes a service-linked role.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_service_linked_role(role_name)
    response = @iam_client.delete_service_linked_role(role_name: role_name)
    task_id = response.deletion_task_id
    check_deletion_status(role_name, task_id)
  rescue Aws::Errors::ServiceError => e
    handle_deletion_error(e, role_name)
  end

  private
```

```
# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  unless e.code == "NoSuchEntity"
    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Per i dettagli sull'API, DeleteServiceLinkedRoleconsulta AWS SDK for Ruby API Reference.

Eliminazione di un utente

Il seguente esempio di codice mostra come eliminare un utente IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteUserconsulta AWS SDK for Ruby API Reference.

Eliminare una chiave di accesso

Il seguente esempio di codice mostra come eliminare una chiave di accesso IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, disattiva ed elimina le chiavi di accesso.

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

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

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

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

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

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

- Per i dettagli sull'API, consulta la sezione API DeleteAccessKeyReference AWS SDK for Ruby .

Eliminazione di un alias di un account

Il seguente esempio di codice mostra come eliminare l'alias di un account IAM.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, crea ed elimina gli alias degli account.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end
```

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

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

- Per i dettagli sull'API, consulta la sezione DeleteAccountAlias AWS SDK for RubyAPI Reference.

Eliminazione di una policy inline da un utente

Il seguente esempio di codice mostra come eliminare una policy IAM in linea da un utente.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteUserPolicyconsulta AWS SDK for Ruby API Reference.

Scollegamento di una policy da un ruolo

Il seguente esempio di codice mostra come scollegare una policy IAM da un ruolo.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, allega e scollega le politiche relative ai ruoli.

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

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

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

```
    raise
  end

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

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

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

```
    false
  end
end
```

- Per i dettagli sulle API, consulta la sezione AWS SDK for Ruby API [DetachRolePolicy](#)Reference.

Scollegamento di una policy da un utente

Il seguente esempio di codice mostra come scollegare una policy IAM da un utente.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DetachUserPolicyconsulta AWS SDK for Ruby API Reference.

Ottenere una policy

Il seguente esempio di codice mostra come ottenere una policy IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetPolicyconsulta AWS SDK for Ruby API Reference.

Recupero di un ruolo

Il seguente esempio di codice mostra come ottenere un ruolo IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetRoleconsulta AWS SDK for Ruby API Reference.

Ottenimento di utente

I seguenti esempi di codice mostrano come ottenere un ruolo IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, GetUserconsulta AWS SDK for Ruby API Reference.

Recupero della policy sulla password dell'account

Il seguente esempio di codice mostra come ottenere la politica relativa alle password degli account IAM.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "IAMPolicyManager"
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    begin
      response = @iam_client.get_account_password_policy
      @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
      raise
    end
  end
end
```

- Per i dettagli sull'API, GetAccountPasswordPolicyconsulta AWS SDK for Ruby API Reference.

Elencare gli IdP SAML

Il seguente esempio di codice mostra come elencare i provider SAML per IAM.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per informazioni dettagliate sull'API, consulta la sezione ListSAMLProviders nella Documentazione di riferimento dell'API AWS SDK for Ruby .

Elencare le chiavi di accesso di un utente

Il seguente esempio di codice mostra come elencare le chiavi di accesso IAM di un utente.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, disattiva ed elimina le chiavi di accesso.

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

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

```
  end

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

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

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

```
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Per i dettagli sull'API, consulta la sezione API ListAccessKeysReference AWS SDK for Ruby .

Elencare gli alias di un account

Il seguente esempio di codice mostra come elencare gli alias degli account IAM.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, crea ed elimina gli alias degli account.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
```

```
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

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

- Per i dettagli sull'API, consulta la sezione ListAccountAliases AWS SDK for RubyAPI Reference.

Elencare i gruppi

Il seguente esempio di codice mostra come elencare i gruppi IAM.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

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

- Per i dettagli sull'API, ListGroupsconsulta AWS SDK for Ruby API Reference.

Elencare le policy inline per un ruolo

Il seguente esempio di codice mostra come elencare le politiche in linea per un ruolo IAM.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, ListRolePoliciesconsulta AWS SDK for Ruby API Reference.

Elencare le policy

Il seguente esempio di codice mostra come elencare le politiche IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, allega e scollega le politiche relative ai ruoli.

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

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

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

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

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

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

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

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

- Per i dettagli sulle API, consulta la sezione AWS SDK for Ruby API ListPoliciesReference.

Elencare le policy collegate a un ruolo

Il seguente esempio di codice mostra come elencare le policy associate a un ruolo IAM.

SDK per Ruby

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Questo modulo di esempio elenca, crea, allega e scollega le politiche relative ai ruoli.

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

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

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

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

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

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

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

- Per i dettagli sulle API, consulta la sezione AWS SDK for Ruby API ListAttachedRolePoliciesReference.

Elencare i ruoli

Il seguente esempio di codice mostra come elencare i ruoli IAM.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, ListRolesconsulta AWS SDK for Ruby API Reference.

Elencare i certificati del server

L'esempio di codice seguente mostra come elencare tutti i certificati del server IAM.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, aggiorna ed elimina i certificati del server.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
```

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

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

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

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

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

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

- Per i dettagli sull'API, consulta la ListServerCertificatessezione AWS SDK for Ruby API Reference.

Elencare gli utenti

Il seguente esempio di codice mostra come elencare gli utenti IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, ListUsersconsulta AWS SDK for Ruby API Reference.

Aggiornamento di un certificato del server

L'esempio di codice seguente mostra come aggiornare un certificato del server IAM.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Elenca, aggiorna ed elimina i certificati del server.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
```

```
                                          server_certificate_name: name,
                                          certificate_body: certificate_body,
                                          private_key: private_key,
                                        })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

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

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

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

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

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
```

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

- Per i dettagli sull'API, consulta la UpdateServerCertificatesezione AWS SDK for Ruby API Reference.

Aggiornamento di un utente

Il seguente esempio di codice mostra come aggiornare un utente IAM.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, UpdateUserconsulta AWS SDK for Ruby API Reference.

Scenari

Creazione di un utente e assunzione di un ruolo

Il seguente esempio di codice mostra come creare un utente e assumere un ruolo.

> **Warning**
>
> Per evitare rischi per la sicurezza, non utilizzare gli utenti IAM per l'autenticazione quando sviluppi software creato ad hoc o lavori con dati reali. Utilizza invece la federazione con un provider di identità come AWS IAM Identity Center.

- Crea un utente che non disponga di autorizzazioni.
- Crea un ruolo che conceda l'autorizzazione per elencare i bucket Amazon S3 per l'account.
- Aggiungi una policy per consentire all'utente di assumere il ruolo.
- Assumi il ruolo ed elenca i bucket S3 utilizzando le credenziali temporanee, quindi ripulisci le risorse.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea un utente IAM che conceda l'autorizzazione per elencare i bucket Amazon S3. L'utente dispone dei diritti soltanto per assumere il ruolo. Dopo aver assunto il ruolo, utilizza le credenziali temporanee per elencare i bucket per l'account.

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

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

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

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

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

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

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Principal: {'AWS': user.arn},
                   Action: "sts:AssumeRole"
                 }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
```

```
                  Effect: "Allow",
                  Action: "s3:ListAllMyBuckets",
                  Resource: "arn:aws:s3:::*"
                }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                  Effect: "Allow",
                  Action: "sts:AssumeRole",
                  Resource: role.arn
                }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
```

```
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

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

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

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
```

```
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

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

```
  end

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

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

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
```

```
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

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

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Esempi di Kinesis con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con Kinesis.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Esempi serverless

Esempi serverless

Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento Kinesis con Lambda utilizzando Ruby.

```
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
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori relativi agli elementi batch di Kinesis con Lambda utilizzando Ruby.

```
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 esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with AWS KMS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Crea una chiave .

Il seguente esempio di codice mostra come creare un AWS KMS key.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, CreateKeyconsulta AWS SDK for Ruby API Reference.

Decrittografa il testo cifrato

Il seguente esempio di codice mostra come decrittografare il testo cifrato che è stato crittografato da una chiave KMS.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta Decrypt in AWS SDK for Ruby API Reference.

Crittografa il testo usando una chiave

Il seguente esempio di codice mostra come crittografare il testo utilizzando una chiave KMS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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*")
```

- Per i dettagli sull'API, consulta Encrypt in AWS SDK for Ruby API Reference.

Ricrittografa il testo cifrato da una chiave all'altra

Il seguente esempio di codice mostra come ricrittografare il testo cifrato da una chiave KMS a un'altra.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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*")
```

- Per i dettagli sull'API, ReEncryptconsulta AWS SDK for Ruby API Reference.

Esempi di Lambda con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with Lambda.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Creazione di una funzione

Il seguente esempio di codice mostra come creare una funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
```

```
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function. This
  #                      must include the file name and the function name.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  #                            code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: "ruby2.7",
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    "LOG_LEVEL" => "info"
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name})
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Per i dettagli sull'API, CreateFunctionconsulta AWS SDK for Ruby API Reference.

Eliminare una funzione

Il seguente esempio di codice mostra come eliminare una funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print "Done!".green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- Per i dettagli sull'API, DeleteFunctionconsulta AWS SDK for Ruby API Reference.

Ottenimento di una funzione

L'esempio di codice seguente mostra come ottenere una funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Per i dettagli sull'API, GetFunctionconsulta AWS SDK for Ruby API Reference.

Richiamo di una funzione

Il seguente esempio di codice mostra come richiamare una funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name}
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Per informazioni dettagliate sulle API, consulta Invoke nella Documentazione di riferimento delle API AWS SDK for Ruby .

Elencare le funzioni

L'esempio di codice seguente mostra come elencare le funzioni Lambda.

SDK per Ruby

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

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

- Per i dettagli sull'API, ListFunctionsconsulta AWS SDK for Ruby API Reference.

Aggiornamento del codice della funzione

L'esempio di codice seguente mostra come aggiornare il codice della funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.

  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

- Per i dettagli sull'API, UpdateFunctionCodeconsulta AWS SDK for Ruby API Reference.

Aggiornamento della configurazione della funzione

L'esempio di codice seguente mostra come aggiornare la configurazione della funzione Lambda.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       "LOG_LEVEL" => log_level
                                                     }
                                                   }
                                                 })
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
 do |w|
      w.max_attempts = 5
      w.delay = 5
```

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

- Per i dettagli sull'API, UpdateFunctionConfigurationconsulta AWS SDK for Ruby API Reference.

Scenari

Nozioni di base sulle funzioni

L'esempio di codice seguente mostra come:

- Crea un ruolo IAM e una funzione Lambda, quindi carica il codice del gestore.
- Richiamare la funzione con un singolo parametro e ottenere i risultati.
- Aggiorna il codice della funzione e configuralo con una variabile di ambiente.
- Richiamare la funzione con nuovi parametri e ottenere i risultati. Visualizza il log di esecuzione restituito.
- Elenca le funzioni dell'account, quindi elimina le risorse.

Per ulteriori informazioni sull'utilizzo di Lambda, consulta Creare una funzione Lambda con la console.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Configura le autorizzazioni IAM prerequisite per una funzione Lambda in grado di scrivere log.

```
  # Get an AWS Identity and Access Management (IAM) role.
  #
```

```
  # @param iam_role_name: The name of the role to retrieve.
  # @param action: Whether to create or destroy the IAM apparatus.
  # @return: The IAM role.
  def manage_iam(iam_role_name, action)
    role_policy = {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Effect': "Allow",
          'Principal': {
            'Service': "lambda.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }
    case action
    when "create"
      role = $iam_client.create_role(
        role_name: iam_role_name,
        assume_role_policy_document: role_policy.to_json
      )
      $iam_client.attach_role_policy(
        {
          policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
          role_name: iam_role_name
        }
      )
      $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
        w.max_attempts = 5
        w.delay = 5
      end
      @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
      @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
 fully.")
      sleep(10)
      return role, role_policy.to_json
    when "destroy"
      $iam_client.detach_role_policy(
        {
          policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
          role_name: iam_role_name
```

```
        }
      )
      $iam_client.delete_role(
        role_name: iam_role_name
      )
      @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
    else
      raise "Incorrect action provided. Must provide 'create' or 'destroy'"
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating role or attaching policy:\n
 #{e.message}")
  end
```

Definisci un gestore Lambda che incrementa un numero fornito come parametro di chiamata.

```
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
```

Comprimi la funzione Lambda in un pacchetto di implementazione:

```
 # Creates a Lambda deployment package in .zip format.
 # This zip can be passed directly as a string to Lambda when creating the
function.
 #
 # @param source_file: The name of the object, without suffix, for the Lambda file
and zip.
 # @return: The deployment package.
 def create_deployment_package(source_file)
   Dir.chdir(File.dirname(__FILE__))
   if File.exist?("lambda_function.zip")
     File.delete("lambda_function.zip")
     @logger.debug("Deleting old zip: lambda_function.zip")
   end
   Zip::File.open("lambda_function.zip", create: true) {
     |zipfile|
     zipfile.add("lambda_function.rb", "#{source_file}.rb")
   }
   @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
   File.read("lambda_function.zip").to_s
 rescue StandardError => e
   @logger.error("There was an error creating deployment package:\n #{e.message}")
 end
```

Crea una nuova funzione Lambda.

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

```
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: "ruby2.7",
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    "LOG_LEVEL" => "info"
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name})
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Richiama la tua funzione Lambda con parametri di runtime facoltativi.

```
  # 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
```

Aggiorna la configurazione della funzione Lambda per inserire una nuova variabile di ambiente.

```
  # 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
```

Aggiorna il codice della funzione Lambda con un pacchetto di implementazione diverso contenente codice diverso.

```
  # 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
```

Elenca tutte le funzioni Lambda esistenti utilizzando l'impaginatore integrato.

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

Elimina una funzione Lambda specifica.

```
  # 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
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Esempi serverless

Richiamare una funzione Lambda da un trigger Kinesis

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso Kinesis. La funzione recupera il payload Kinesis, lo decodifica da Base64 e registra il contenuto del record.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento Kinesis con Lambda utilizzando Ruby.

```
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
```

Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento SNS con Lambda utilizzando Ruby.

```
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
```

Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SQS con Lambda tramite Ruby.

```
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
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da un flusso Kinesis. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori relativi agli elementi batch di Kinesis con Lambda utilizzando Ruby.

```
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
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di SQS con Lambda tramite Ruby.

```
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
```

Esempi di Amazon Polly con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Polly. AWS SDK for Ruby

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Rendi disponibili le voci per la sintesi

Il seguente esempio di codice mostra come rendere disponibili le voci di Amazon Polly per la sintesi.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: "en-US")

  resp.voices.each do |v|
    puts v.name
    puts "  " + v.gender
    puts
  end
rescue StandardError => ex
  puts "Could not get voices"
  puts "Error message:"
  puts ex.message
end
```

- Per i dettagli sull'API, DescribeVoicesconsulta AWS SDK for Ruby API Reference.

Elenca i lessici di pronuncia

Il seguente esempio di codice mostra come elencare i lessici di pronuncia di Amazon Polly.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
```

```
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:" + l.attributes.alphabet
    puts "  Language:" + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts "Could not get lexicons"
  puts "Error message:"
  puts ex.message
end
```

- Per i dettagli sull'API, ListLexiconsconsulta AWS SDK for Ruby API Reference.

Sintetizza il parlato dal testo

Il seguente esempio di codice mostra come sintetizzare il parlato dal testo con Amazon Polly.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-polly"  # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts "You must supply a filename"
    exit 1
  end
```

```
  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: " + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split(".")
  first_part = parts[0]
  mp3_file = first_part + ".mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: " + mp3_file
rescue StandardError => ex
  puts "Got error:"
  puts "Error message:"
  puts ex.message
end
```

- Per i dettagli sull'API, SynthesizeSpeechconsulta AWS SDK for Ruby API Reference.

Esempi di Amazon RDS con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon RDS. AWS SDK for Ruby

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Creare uno snapshot di un'istanza database

Il seguente esempio di codice mostra come creare uno snapshot di un'istanza database Amazon RDS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per informazioni dettagliate sull'API, consulta CreateDBSnapshot nella Documentazione di riferimento dell'API AWS SDK for Ruby .

Descrizione delle istanze database

Il seguente esempio di codice mostra come descrivere le istanze database di Amazon RDS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per informazioni dettagliate sull'API, consulta DescribeDBInstances nella Documentazione di riferimento delle API di AWS SDK for Ruby .

Descrivere i gruppi di parametri database

Il seguente esempio di codice mostra come descrivere i gruppi di parametri di Amazon RDS DB.

SDK per Ruby

> **Note**
>
> C'è altro da sapere. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, consulta DescribeDB ParameterGroups in AWS SDK for Ruby API Reference.

Descrivere i parametri in un gruppo di parametri database

Il seguente esempio di codice mostra come descrivere i parametri in un gruppo di parametri Amazon RDS DB.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per informazioni sull'API, consulta DescribeDBParameters nella Documentazione di riferimento dell'API AWS SDK for Ruby .

Descrivere gli snapshot delle istanze database

Il seguente esempio di codice mostra come descrivere gli snapshot delle istanze DB di Amazon RDS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per informazioni dettagliate sull'API, consulta DescribeDBSnapshots nella Documentazione di riferimento dell'API AWS SDK for Ruby .

Esempi di Amazon S3 che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon S3. AWS SDK for Ruby

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Aggiunta di regole CORS a un bucket

Il seguente esempio di codice mostra come aggiungere regole CORS (Cross-Origin Resource Sharing) a un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, PutBucketCorsconsulta AWS SDK for Ruby API Reference.

Aggiunta di una policy a un bucket

Il seguente esempio di codice mostra come aggiungere una policy a un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Per i dettagli sull'API, PutBucketPolicyconsulta AWS SDK for Ruby API Reference.

Copia di un oggetto da un bucket a un altro

Il seguente esempio di codice mostra come copiare un oggetto S3 da un bucket a un altro.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Copia un oggetto.

```
require "aws-sdk-s3"
```

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

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

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

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

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

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

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copia un oggetto e aggiungi la crittografia lato server all'oggetto di destinazione.

```
require "aws-sdk-s3"

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

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

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

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
```

```
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CopyObjectconsulta AWS SDK for Ruby API Reference.

Creazione di un bucket

Il seguente esempio di codice mostra come creare un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
```

```
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, CreateBucketconsulta AWS SDK for Ruby API Reference.

Eliminazione di regole CORS da un bucket

Il seguente esempio di codice mostra come eliminare le regole CORS da un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteBucketCorsconsulta AWS SDK for Ruby API Reference.

Eliminazione di una policy da un bucket

Il seguente esempio di codice mostra come eliminare una policy da un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteBucketPolicyconsulta AWS SDK for Ruby API Reference.

Eliminazione di un bucket vuoto

Il seguente esempio di codice mostra come eliminare un bucket S3 vuoto.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteBucketconsulta AWS SDK for Ruby API Reference.

Eliminazione di più oggetti

Il seguente esempio di codice mostra come eliminare più oggetti da un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, DeleteObjectsconsulta AWS SDK for Ruby API Reference.

Determinazione del tipo di esistenza e contenuto di un oggetto

Il seguente esempio di codice mostra come determinare l'esistenza e il tipo di contenuto di un oggetto in un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"

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

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

```
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, HeadObjectconsulta AWS SDK for Ruby API Reference.

Recupero di regole CORS per un bucket

Il seguente esempio di codice mostra come ottenere regole CORS (Cross-Origin Resource Sharing) per un bucket S3.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- Per i dettagli sull'API, GetBucketCorsconsulta AWS SDK for Ruby API Reference.

Recupero di un oggetto da un bucket

Il seguente esempio di codice mostra come leggere i dati da un oggetto in un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Recupera un oggetto.

```
require "aws-sdk-s3"

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

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

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

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Recupera un oggetto e segnala lo stato di crittografia lato server.

```
require "aws-sdk-s3"

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

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

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

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

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

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
```

```
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, GetObjectconsulta AWS SDK for Ruby API Reference.

Recupero della policy per un bucket

Il seguente esempio di codice mostra come ottenere la policy per un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end
```

```
end
```

- Per i dettagli sull'API, GetBucketPolicyconsulta AWS SDK for Ruby API Reference.

Elenco di bucket

Il seguente esempio di codice mostra come elencare i bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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__
```

- Per i dettagli sull'API, ListBucketsconsulta AWS SDK for Ruby API Reference.

Elenco di oggetti in un bucket

Il seguente esempio di codice mostra come elencare gli oggetti in un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

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

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
```

```
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la ListObjectsversione V2 in AWS SDK for Ruby API Reference.

Impostazione della configurazione del sito Web per un bucket

Il seguente esempio di codice mostra come impostare la configurazione del sito Web per un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

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

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

```
run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, PutBucketWebsiteconsulta AWS SDK for Ruby API Reference.

Caricamento di un oggetto in un bucket

Il seguente esempio di codice mostra come caricare un oggetto in un bucket S3.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Carica un file utilizzando un caricamento gestito (Object.upload_file).

```
require "aws-sdk-s3"

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

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

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
```

```
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Carica un file utilizzando Object.put.

```
require "aws-sdk-s3"

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

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

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Carica un file utilizzando Object.put e aggiungi la crittografia lato server.

```
require "aws-sdk-s3"

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

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

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
```

```
  object_content = "This is my super-secret content."
  encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, PutObjectconsulta AWS SDK for Ruby API Reference.

Scenari

Creazione di un URL prefirmato

Il seguente esempio di codice mostra come creare un URL predefinito per Amazon S3 e caricare un oggetto.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
```

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

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

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Nozioni di base su bucket e oggetti

L'esempio di codice seguente mostra come:

- Crea un bucket e carica un file in tale bucket.
- Scaricare un oggetto da un bucket.
- Copiare un oggetto in una sottocartella in un bucket.
- Elencare gli oggetti in un bucket.

- Elimina il bucket e tutti gli oggetti in esso contenuti.

SDK per Ruby

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

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

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end
```

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

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

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

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

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

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

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

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

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

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

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

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

Esempi di Amazon SES con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con Amazon SES.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Ottenimento dello stato di un'identità

L'esempio di codice seguente mostra come ottenere lo stato di un'identità di Amazon SES.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")
```

```
# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Per i dettagli sull'API, GetIdentityVerificationAttributesconsulta AWS SDK for Ruby API Reference.

Elenco di tutte le identità

Il seguente esempio di codice mostra come elencare le identità di Amazon SES.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")
```

```
# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Per i dettagli sull'API, ListIdentitiesconsulta AWS SDK for Ruby API Reference.

Invio di e-mail

Il seguente esempio di codice mostra come inviare e-mail con Amazon SES.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = "sender@example.com"

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
```

```
recipient = "recipient@example.com"

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = "Amazon SES test (AWS SDK for Ruby)"

# The HTML body of the email.
htmlbody =
  "<h1>Amazon SES test (AWS SDK for Ruby)</h1>"\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  "AWS SDK for Ruby</a>."

# The email body for recipients with non-HTML email clients.
textbody = "This email was sent with Amazon SES using the AWS SDK for Ruby."

# Specify the text encoding scheme.
encoding = "UTF-8"

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: "us-west-2")

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
```

```
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
    )

  puts "Email sent to " + recipient

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

- Per i dettagli sull'API, SendEmailconsulta AWS SDK for Ruby API Reference.

Verifica di un'identità e-mail

L'esempio di codice seguente mostra come verificare un'identità e-mail con Amazon SES.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-ses"  # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
```

```
ses = Aws::SES::Client.new(region: "us-west-2")

# Try to verify email address.
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts "Email sent to " + recipient

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

- Per i dettagli sull'API, VerifyEmailIdentityconsulta AWS SDK for Ruby API Reference.

Esempi di API Amazon SES v2 con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l'API AWS SDK for Ruby with Amazon SES v2.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Invia un'e-mail

L'esempio di codice seguente mostra come inviare un'e-mail dell'API Amazon SES v2.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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)
```

- Per i dettagli sull'API, SendEmailconsulta AWS SDK for Ruby API Reference.

Esempi di Amazon SNS che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con Amazon SNS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni
- Esempi serverless

Azioni

Creazione di un argomento

Il seguente esempio di codice mostra come creare un argomento Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for Ruby API Reference.

Come elencare i sottoscrittori di un argomento

Il seguente esempio di codice mostra come recuperare l'elenco degli abbonati di un argomento Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, ListSubscriptionsconsulta AWS SDK for Ruby API Reference.

Come elencare gli argomenti

Il seguente esempio di codice mostra come elencare gli argomenti di Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
```

```
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, ListTopicsconsulta AWS SDK for Ruby API Reference.

Pubblicazione in un argomento

Il seguente esempio di codice mostra come pubblicare messaggi su un argomento di Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for Ruby .

Impostazione degli attributi degli argomenti

Il seguente esempio di codice mostra come impostare gli attributi degli argomenti di Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for Ruby API Reference.

Sottoscrizione di un indirizzo e-mail a un argomento

Il seguente esempio di codice mostra come iscrivere un indirizzo e-mail a un argomento di Amazon SNS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
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
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for Ruby .

Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon SNS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento SNS con Lambda utilizzando Ruby.

```
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
```

Esempi di Amazon SQS con SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby con Amazon SQS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

Azioni

Modifica la visibilità del timeout dei messaggi

Il seguente esempio di codice mostra come modificare la visibilità del timeout di un messaggio Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-sqs"  # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: "us-west-2")
```

```
begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not be visible for 30 seconds after
 first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."

rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{receive_queue_name}', as it
 does not exist."
end
```

- Per i dettagli sull'API, ChangeMessageVisibilityconsulta AWS SDK for Ruby API Reference.

Crea una coda

Il seguente esempio di codice mostra come creare una coda Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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__
```

- Per i dettagli sull'API, CreateQueueconsulta AWS SDK for Ruby API Reference.

Elimina una coda

Il seguente esempio di codice mostra come eliminare una coda Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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)
```

- Per i dettagli sull'API, DeleteQueueconsulta AWS SDK for Ruby API Reference.

Elencare code

Il seguente esempio di codice mostra come elencare le code Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, ListQueuesconsulta AWS SDK for Ruby API Reference.

Ricevi messaggi da una coda

Il seguente esempio di codice mostra come ricevere messaggi da una coda Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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__
```

- Per i dettagli sull'API, ReceiveMessageconsulta AWS SDK for Ruby API Reference.

Invia un batch di messaggi a una coda

Il seguente esempio di codice mostra come inviare un batch di messaggi a una coda Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
```

```
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  entries = [
    {
      id: "Message1",
      message_body: "This is the first message."
    },
    {
      id: "Message2",
      message_body: "This is the second message."
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)
```

```
  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts "Messages sent."
  else
    puts "Messages not sent."
  end
end
```

- Per i dettagli sull'API, SendMessageBatchconsulta AWS SDK for Ruby API Reference.

Inviare un messaggio a una coda

Il seguente esempio di codice mostra come inviare un messaggio a una coda Amazon SQS.

SDK per Ruby

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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__
```

- Per i dettagli sull'API, SendMessageconsulta AWS SDK for Ruby API Reference.

Esempi serverless

Richiamo di una funzione Lambda da un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da una coda SQS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Utilizzo di un evento SQS con Lambda tramite Ruby.

```
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
```

Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

Il seguente esempio di codice mostra come implementare una risposta batch parziale per le funzioni Lambda che ricevono eventi da una coda SQS. La funzione riporta gli errori degli elementi batch nella risposta, segnalando a Lambda di riprovare tali messaggi in un secondo momento.

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Segnalazione di errori di elementi batch di SQS con Lambda tramite Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK for Ruby with AWS STS.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Assunzione di un ruolo

Il seguente esempio di codice mostra come assumere un ruolo con AWS STS.

SDK per Ruby

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice 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
```

- Per i dettagli sull'API, AssumeRoleconsulta AWS SDK for Ruby API Reference.

WorkDocs Esempi di Amazon che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il AWS SDK for Ruby con Amazon WorkDocs.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- Azioni

Azioni

Descrivi il contenuto della cartella principale

Il seguente esempio di codice mostra come descrivere il contenuto WorkDocs della cartella principale di Amazon.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
 # Retrieves the root folder for a user by email
 # @param users [Array<Types::User>] A list of users selected from API response
 # @param user_email [String] The email of the user.
 def get_user_folder(users, user_email)
   user = users.find { |user| user.email_address == user_email }
   if user
     user.root_folder_id
   else
     @logger.error "Could not get root folder for user with email address
#{user_email}"
     exit(1)
   end
 end

 # Describes the contents of a folder
 # @param [String] folder_id - The Id of the folder to describe.
 def describe_folder_contents(folder_id)
   resp = @client.describe_folder_contents({
                                             folder_id: folder_id, # required
                                             sort: "NAME", # accepts DATE, NAME
                                             order: "ASCENDING", # accepts
ASCENDING, DESCENDING
                                           })
   resp.documents.each do |doc|
     md = doc.latest_version_metadata
     @logger.info "Name:          #{md.name}"
     @logger.info "Size (bytes):  #{md.size}"
```

```
    @logger.info "Last modified: #{doc.modified_timestamp}"
    @logger.info "Doc ID:        #{doc.id}"
    @logger.info "Version ID:    #{md.id}"
    @logger.info ""
  end
rescue Aws::WorkDocs::Errors::ServiceError => e
  @logger.error "Error listing folder contents: #{e.message}"
  exit(1)
end
```

- Per i dettagli sull'API, DescribeRootFolderconsulta AWS SDK for Ruby API Reference.

Descrivi utenti

Il seguente esempio di codice mostra come descrivere WorkDocs gli utenti Amazon.

SDK per Ruby

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# Describes users within an organization
# @param [String] org_id: The ID of the org.
def describe_users(org_id)
  resp = @client.describe_users({
                                  organization_id: org_id,
                                  include: "ALL", # accepts ALL, ACTIVE_PENDING
                                  order: "ASCENDING", # accepts ASCENDING,
DESCENDING
                                  sort: "USER_NAME", # accepts USER_NAME,
FULL_NAME, STORAGE_LIMIT, USER_STATUS, STORAGE_USED
                                })
  resp.users.each do |user|
    @logger.info "First name:  #{user.given_name}"
    @logger.info "Last name:   #{user.surname}"
    @logger.info "Email:       #{user.email_address}"
    @logger.info "Root folder: #{user.root_folder_id}"
    @logger.info ""
```

```
    end
    resp.users
  rescue Aws::WorkDocs::Errors::ServiceError => e
    @logger.error "AWS WorkDocs Service Error: #{e.message}"
    exit(1)
  end
```

- Per i dettagli sull'API, DescribeUsersconsulta AWS SDK for Ruby API Reference.

Esempi interservizi che utilizzano SDK for Ruby

Le seguenti applicazioni di esempio utilizzano il AWS SDK for Ruby per funzionare su più applicazioni Servizi AWS.

Gli esempi trasversali mirano a un livello avanzato di esperienza per aiutarti a iniziare a creare applicazioni.

Esempi

- Crea un'applicazione che analizza il feedback dei clienti e sintetizza l'audio

Crea un'applicazione che analizza il feedback dei clienti e sintetizza l'audio

SDK per Ruby

Questa applicazione di esempio analizza e archivia le schede di feedback dei clienti. In particolare, soddisfa l'esigenza di un hotel fittizio a New York City. L'hotel riceve feedback dagli ospiti in varie lingue sotto forma di schede di commento fisiche. Tale feedback viene caricato nell'app tramite un client Web. Dopo aver caricato l'immagine di una scheda di commento, vengono eseguiti i seguenti passaggi:

- Il testo viene estratto dall'immagine utilizzando Amazon Textract.
- Amazon Comprehend determina il sentiment del testo estratto e la sua lingua.
- Il testo estratto viene tradotto in inglese utilizzando Amazon Translate.
- Amazon Polly sintetizza un file audio dal testo estratto.

L'app completa può essere implementata con AWS CDK. Per il codice sorgente e le istruzioni di distribuzione, consulta il progetto in GitHub.

Servizi utilizzati in questo esempio

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

Sicurezza per AWS SDK for Ruby

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In quanto cliente AWS , è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza. La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud.

Security of the Cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce tutti i servizi offerti nel AWS Cloud e della fornitura di servizi che è possibile utilizzare in modo sicuro. La nostra responsabilità in AWS materia di sicurezza è la massima priorità e l'efficacia della nostra sicurezza viene regolarmente testata e verificata da revisori di terze parti nell'ambito dei Programmi di AWS conformità.

Sicurezza nel cloud: la responsabilità dell'utente è determinata dai dati Servizio AWS utilizzati e da altri fattori, tra cui la sensibilità dei dati, i requisiti dell'organizzazione e le leggi e i regolamenti applicabili.

Argomenti

Protezione dei dati in AWS SDK for Ruby

Il modello di responsabilità AWS condivisa di si applica alla protezione dei dati in. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla privacy dei dati, vedi Domande frequenti sulla privacy

dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog AWS Shared Responsibility Model and GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori o Servizi AWS utilizzi la console, l'API o gli SDK. AWS CLI AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Identity and Access Management per AWS SDK for Ruby

AWS Identity and Access Management (IAM) è un servizio Amazon Web Services (AWS) che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse Servizi AWS. IAM è un servizio Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Per utilizzare AWS SDK for Ruby per accedere, sono AWS necessari un AWS AWS account e delle credenziali. Per rafforzare la sicurezza dell'account AWS , ti consigliamo di utilizzare un utente IAM per fornire le credenziali di accesso anziché utilizzare le credenziali dell'account AWS .

Per i dettagli sull'utilizzo di IAM, consulta IAM.

Per una panoramica sugli utenti IAM e sul motivo per cui sono importanti per la sicurezza dell'account, consulta Credenziali di sicurezza AWS in Riferimenti generali di Amazon Web Services.

AWS SDK for Ruby segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web AWS Services () che supporta. Per informazioni sulla Servizio AWS sicurezza, consulta la pagina della documentazione Servizio AWS sulla sicurezza e quelle Servizi AWS che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Convalida della conformità per AWS SDK for Ruby

AWS SDK for Ruby segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web AWS Services () che supporta. Per informazioni sulla Servizio AWS sicurezza, consulta la pagina della documentazione Servizio AWS sulla sicurezza e quelle Servizi AWS che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

La sicurezza e la conformità dei servizi Amazon Web Services (AWS) vengono valutate da revisori di terze parti nell'ambito di più programmi di AWS conformità. Questi includono SOC, PCI, FedRAMP, HIPAA e altri. AWS fornisce un elenco aggiornato di frequente dei programmi di conformità specifici Servizi AWS nella pagina AWS Services in Scope by Compliance Program.

I report di audit di terze parti possono essere scaricati utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact.

Per ulteriori informazioni sui programmi di AWS conformità, consulta Programmi di AWS conformità.

La tua responsabilità di conformità quando utilizzi AWS SDK for Ruby per Servizio AWS accedere a un è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità dell'organizzazione e dalle leggi e dai regolamenti applicabili. Se l'uso di un Servizio AWS è soggetto alla conformità a standard come HIPAA, PCI o FedRAMP, fornisce risorse per aiutare a: AWS

- Guide introduttive su sicurezza e conformità: guide all'implementazione che illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e sulla conformità. AWS
- Whitepaper sull'architettura per la sicurezza e la conformità HIPAA: un white paper che descrive in che modo le aziende possono utilizzare per creare applicazioni conformi allo standard HIPAA. AWS

- AWS Risorse per la conformità: una raccolta di cartelle di lavoro e guide che potrebbero riguardare il settore e la località in cui operi.
- AWS Config: un servizio che valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub: una visione completa dello stato di sicurezza interno AWS che consente di verificare la conformità agli standard e alle best practice del settore della sicurezza.

Resilienza per AWS SDK for Ruby

L'infrastruttura globale di Amazon Web Services (AWS) è costruita attorno a Regioni AWS zone di disponibilità.

Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti.

Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni sulle zone di disponibilità, vedere Global Regioni AWS Infrastructure.AWS

AWS SDK for Ruby segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web AWS Services () che supporta. Per informazioni sulla Servizio AWS sicurezza, consulta la pagina della documentazione Servizio AWS sulla sicurezza e quelle Servizi AWS che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Sicurezza dell'infrastruttura per AWS SDK for Ruby

AWS SDK for Ruby segue il modello di responsabilità condivisa attraverso i servizi specifici di Amazon Web AWS Services () che supporta. Per informazioni sulla Servizio AWS sicurezza, consulta la pagina della documentazione Servizio AWS sulla sicurezza e quelle Servizi AWS che rientrano nell'ambito delle iniziative di AWS conformità previste dal programma di conformità.

Per informazioni sui processi AWS di sicurezza, consulta il white paper AWS: Panoramica dei processi di sicurezza.

Applicazione di una versione TLS minima nell' AWS SDK for Ruby

La comunicazione tra AWS SDK for AWS Ruby e è protetta tramite Secure Sockets Layer (SSL) o Transport Layer Security (TLS). Tutte le versioni di SSL e le versioni di TLS precedenti alla 1.2 presentano vulnerabilità che possono compromettere la sicurezza delle comunicazioni. AWS Per questo motivo, dovresti assicurarti di utilizzare l' AWS SDK per Ruby con una versione di Ruby che supporti TLS versione 1.2 o successiva.

Ruby utilizza la libreria OpenSSL per proteggere le connessioni HTTP. Le versioni supportate di Ruby (1.9.3 e successive) installate tramite gestori di pacchetti di sistema (e altri) `yumapt`, un programma di installazione ufficiale o i gestori Ruby (rbenv, RVM e altri) in genere incorporano OpenSSL 1.0.1 o versioni successive, che supporta TLS 1.2.

Se utilizzato con una versione supportata di Ruby con OpenSSL 1.0.1 o successiva, l'SDK for AWS Ruby preferisce TLS 1.2 e utilizza l'ultima versione di SSL o TLS supportata sia dal client che dal server. Si Servizi AWS tratta sempre almeno di TLS 1.2 per. (L'SDK utilizza la `Net::HTTP` classe Ruby con.) `use_ssl=true`

Verifica della versione OpenSSL

Per assicurarti che l'installazione di Ruby utilizzi OpenSSL 1.0.1 o versione successiva, inserisci il seguente comando.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Un modo alternativo per ottenere la versione OpenSSL è interrogare `openssl` direttamente l'eseguibile. Innanzitutto, individua l'eseguibile appropriato utilizzando il seguente comando.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

L'output dovrebbe `--with-openssl-dir=/path/to/openssl` indicare la posizione dell'installazione di OpenSSL. Prendi nota di questo percorso. Per verificare la versione di OpenSSL, inserisci i seguenti comandi.

```
cd /path/to/openssl
bin/openssl version
```

Quest'ultimo metodo potrebbe non funzionare con tutte le installazioni di Ruby.

Aggiornamento del supporto TLS

Se la versione di OpenSSL utilizzata dall'installazione di Ruby è precedente alla 1.0.1, aggiorna l'installazione di Ruby o OpenSSL utilizzando il gestore dei pacchetti di sistema, il programma di installazione di Ruby o il gestore di Ruby, come descritto nella guida all'installazione di Ruby. Se stai installando Ruby dal codice sorgente, installa prima l'ultima versione di OpenSSL e poi passa durante l'esecuzione. `--with-openssl-dir=/path/to/upgraded/openssl ./configure`

Migrazione del client di crittografia Amazon S3

Questo argomento mostra come migrare le applicazioni dalla versione 1 (V1) del client di crittografia Amazon Simple Storage Service (Amazon S3) alla versione 2 (V2) e garantire la disponibilità delle applicazioni durante tutto il processo di migrazione.

Panoramica sulla migrazione

Questa migrazione avviene in due fasi:

1. Aggiorna i client esistenti per leggere nuovi formati. Innanzitutto, distribuisci una versione aggiornata dell' AWS SDK for Ruby nella tua applicazione. Ciò consentirà ai client di crittografia V1 esistenti di decrittografare gli oggetti scritti dai nuovi client V2. Se l'applicazione utilizza più AWS SDK, è necessario aggiornare ogni SDK separatamente.

2. Esegui la migrazione dei client di crittografia e decrittografia alla versione 2. Una volta che tutti i client di crittografia V1 sono in grado di leggere nuovi formati, è possibile migrare i client di crittografia e decrittografia esistenti alle rispettive versioni V2.

Aggiorna i client esistenti per leggere nuovi formati

Il client di crittografia V2 utilizza algoritmi di crittografia che le versioni precedenti del client non supportano. Il primo passo della migrazione consiste nell'aggiornare i client di decrittografia V1 all'ultima versione dell'SDK. Dopo aver completato questo passaggio, i client V1 dell'applicazione saranno in grado di decrittografare gli oggetti crittografati dai client di crittografia V2. Vedi i dettagli di seguito per ogni versione principale dell' AWS SDK for Ruby.

Aggiorna AWS SDK per Ruby versione 3

La versione 3 è l'ultima versione dell' AWS SDK per Ruby. Per completare questa migrazione, è necessario utilizzare la versione 1.76.0 o successiva del gem. `aws-sdk-s3`

Installazione dalla riga di comando

Per i progetti che installano la `aws-sdk-s3` gem, usa l'opzione version per verificare che sia installata la versione minima di 1.76.0.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Usare Gemfiles

Per i progetti che utilizzano un Gemfile per gestire le dipendenze, imposta la versione minima del `aws-sdk-s3` gem su 1.76.0. Per esempio:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Modifica il tuo Gemfile.
2. Esegui `bundle update aws-sdk-s3`. Per verificare la tua versione, `bundle info aws-sdk-s3` esegui.

Aggiorna AWS SDK per Ruby versione 2

La versione 2 dell' AWS SDK for Ruby entrerà in modalità manutenzione il 21 novembre 2021. Per completare questa migrazione, è necessario utilizzare la versione 2.11.562 o successiva del gem aws-sdk.

Installazione dalla riga di comando

Per i progetti che installano il `aws-sdk` gem, dalla riga di comando, usa l'opzione version per verificare che sia installata la versione minima di 2.11.562.

```
gem install aws-sdk -v '>= 2.11.562'
```

Usare Gemfiles

Per i progetti che utilizzano un Gemfile per gestire le dipendenze, imposta la versione minima del `aws-sdk` gem su 2.11.562. Per esempio:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Modifica il tuo Gemfile. Se hai un file Gemfile.lock, eliminalo o aggiornalo.

2. Esegui `bundle update aws-sdk`. Per verificare la tua versione, esegui. `bundle info aws-sdk`

Migra i client di crittografia e decrittografia alla versione 2

Dopo aver aggiornato i client per leggere i nuovi formati di crittografia, è possibile aggiornare le applicazioni ai client di crittografia e decrittografia V2. I passaggi seguenti mostrano come migrare correttamente il codice dalla V1 alla V2.

Prima di aggiornare il codice per utilizzare il client di crittografia V2, assicurati di aver seguito i passaggi precedenti e di utilizzare la versione `aws-sdk-s3` gem 2.11.562 o successiva.

> **Note**
>
> Quando decifrate con AES-GCM, leggete l'intero oggetto fino alla fine prima di iniziare a utilizzare i dati decrittografati. Questo serve a verificare che l'oggetto non sia stato modificato da quando è stato crittografato.

Configurazione dei client di crittografia V2

EncryptionV2: :Client richiede una configurazione aggiuntiva. Per informazioni dettagliate sulla configurazione, consulta la documentazione di EncryptionV2: :Client o gli esempi forniti più avanti in questo argomento.

1. Il metodo key wrap e l'algoritmo di crittografia del contenuto devono essere specificati nella costruzione del client. Quando ne crei uno nuovo`EncryptionV2::Client`, devi fornire valori per `key_wrap_schema` e`content_encryption_schema`.

`key_wrap_schema`- Se si utilizza AWS KMS, questa opzione deve essere impostata su:`kms_context`. Se si utilizza una chiave simmetrica (AES), questa deve essere impostata su. `:aes_gcm` Se si utilizza una chiave asimmetrica (RSA), questa deve essere impostata su. `:rsa_oaep_sha1`

`content_encryption_schema`- Deve essere impostato su:aes_gcm_no_padding.

2. security_profile deve essere specificato nella costruzione del client. Quando ne crei uno nuovo`EncryptionV2::Client`, devi fornire un valore per. `security_profile` Il parametro security_profile determina il supporto per la lettura di oggetti scritti utilizzando la vecchia V1.

`Encryption::Client` Esistono due valori: :v2 e:v2_and_legacy. Per supportare la migrazione, imposta su:v2_and_legacy*`security_profile`*. Utilizza:v2 solo per lo sviluppo di nuove applicazioni.

3. AWS KMS key L'ID viene applicato per impostazione predefinita. Nella versione 1`Encryption::Client`, il file `kms_key_id` utilizzato per creare il client non veniva fornito a. AWS KMS `Decrypt call` AWS KMS può ottenere queste informazioni dai metadati e aggiungerle al blob di testo cifrato simmetrico. Nella V2, E`NCryptionV2: :Client`, kms_key_id viene passato alla chiamata Decrypt e la chiamata fallisce se non corrisponde alla chiave usata per crittografare l' AWS KMS oggetto. Se in precedenza il codice non si basava sull'impostazione di uno specifico, impostalo alla creazione del client o impostato sulle chiamate. `kms_key_id` `kms_key_id: :kms_allow_decrypt_with_any_cmk` `kms_allow_decrypt_with_any_cmk: true` `get_object`

Esempio: utilizzo di una chiave simmetrica (AES)

Premigrazione

```
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)
```

Post-migrazione

```
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
```

Esempio: utilizzo AWS KMS con kms_key_id

Premigrazione

```
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)
```

Post-migrazione

```
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
```

Esempio: utilizzo AWS KMS senza kms_key_id

Premigrazione

```
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)
```

Post-migrazione

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data)  # No changes
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:
 true) # To allow decrypting with any cmk
```

Alternativa post-migrazione

Se leggi e decrittografi (mai scrivere e crittografare) oggetti utilizzando il client di crittografia S2, usa questo codice.

```
client = Aws::S3::EncryptionV2::Client.new(
```

```
  kms_key_id: :kms_allow_decrypt_with_any_cmk, # set kms_key_id to allow all get_object
 requests to use any cmk
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
 the V1 encryption client
)
resp = client.get_object(bucket: bucket, key: key) # No change
```

Cronologia dei documenti

La tabella che segue descrive le modifiche importanti apportate a questa guida. Per ricevere notifiche sugli aggiornamenti di questa documentazione, puoi sottoscrivere un feed RSS.

Modifica	Descrizione	Data
Tabella dei contenuti	Sommario aggiornato per rendere più accessibili gli esempi di codice.	1 giugno 2023
Aggiornamenti di	Guida aggiornata per allinears i alle best practice IAM. Per ulteriori informazioni, consulta la sezione Best practice per la sicurezza in IAM Aggiornam enti a Guida introduttiva.	8 maggio 2023
Aggiornamenti generali	Aggiornamento della pagina di benvenuto per le risorse esterne pertinenti. È stata aggiornata anche la versione minima richiesta di Ruby per la v2.3. AWS Key Managemen t ServiceSezioni aggiornate per riflettere gli aggiornamenti terminologici. Informazioni di utilizzo aggiornate sull'utilità REPL per maggiore chiarezza.	8 agosto 2022
Correzione dei link non funzionanti	Corretti i link di esempio non funzionanti. Pagina ridondant e di suggerimenti e trucchi rimossa; reindirizzamento a contenuti di esempio di Amazon EC2. Elenchi inclusi degli esempi di codice	3 agosto 2022

	disponibili GitHub nel repository Code Examples.	
Ottenere informazioni su tutti i gruppi di sicurezza Amazon RDS	Aggiunte note relative al ritiro di	26 luglio 2022
SDK Metrics	Sono state rimosse le informazioni sull'attivazione di SDK Metrics for Enterprise Support, che sono state dichiarate obsolete.	28 gennaio 2022

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.