



Guida per gli sviluppatori

# AWS SDK per Ruby



# AWS SDK per Ruby: Guida per gli sviluppatori

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

---

# Table of Contents

Cos'è l' AWS SDK for Ruby? .....	1
Documentazione e risorse aggiuntive .....	1
Implementazione nel cloud AWS .....	1
Manutenzione e supporto per le versioni principali dell'SDK .....	2
Nozioni di base .....	3
Autenticazione con AWS .....	3
Utilizzo delle credenziali della console .....	3
Utilizzo dell'autenticazione IAM Identity Center .....	3
Ulteriori informazioni di autenticazione .....	5
Installazione dell'SDK .....	6
Prerequisiti .....	6
Installazione dell'SDK .....	6
Creazione di una semplice applicazione .....	7
Scrivere il codice .....	8
Esecuzione del programma .....	9
Nota per gli utenti Windows .....	9
Fasi successive .....	10
Configurazione dei client di servizio .....	11
Precedenza delle impostazioni .....	12
Configurazione del client esternamente .....	12
AWS SDK per variabili di ambiente Ruby .....	13
Configurazione del client in codice .....	14
Aws.config .....	14
Regione AWS .....	15
Ordine di ricerca regionale per la risoluzione .....	16
Come impostare la regione .....	16
Fornitori di credenziali .....	17
Catena di fornitori di credenziali .....	18
Creazione di un token di accesso AWS STS .....	20
Tentativi .....	20
Specificazione del comportamento di ripetizione dei tentativi del client nel codice .....	21
Osservabilità .....	21
Configurazione di un client per un servizio <code>OTelProvider</code> .....	22
Configurazione di un servizio <code>OTelProvider</code> per tutti i client di servizio .....	25

Configurazione di un provider di telemetria personalizzato .....	25
Attributi Span .....	25
HTTP .....	27
Impostazione di un endpoint non standard .....	27
Utilizzo di SDK .....	29
Effettuare richieste Servizio AWS .....	29
Utilizzo dell'utilità REPL .....	30
Prerequisiti .....	30
Configurazione del bundler .....	30
Esecuzione di REPL .....	31
Usare l'SDK con Ruby on Rails .....	31
Eseguire il debug tramite wire trace di un client .....	32
Test con stubbing .....	33
Stubbing delle risposte dei clienti .....	33
Stubbing, errori del client .....	35
Paginazione .....	35
Le risposte paginate sono enumerabili .....	35
Gestione manuale delle risposte impaginate .....	36
Classi di dati paginate .....	36
Waiter .....	36
Invocare un cameriere .....	37
Attendi gli errori .....	37
Configurazione di un cameriere .....	38
Allungare un cameriere .....	38
Esempi di codice .....	40
Aurora .....	41
Nozioni di base .....	41
Auto Scaling .....	42
Nozioni di base .....	41
CloudTrail .....	44
Azioni .....	44
CloudWatch .....	48
Azioni .....	44
Gestore dell'identità per Amazon Cognito .....	61
Nozioni di base .....	41
Amazon Comprehend .....	62

Scenari .....	63
Amazon DocumentDB .....	64
Esempi serverless .....	64
DynamoDB .....	65
Nozioni di base .....	41
Nozioni di base .....	67
Azioni .....	44
Scenari .....	63
Esempi serverless .....	64
Amazon EC2 .....	93
Nozioni di base .....	41
Azioni .....	44
Elastic Beanstalk .....	128
Azioni .....	44
EventBridge .....	133
Scenari .....	63
AWS Glue .....	151
Nozioni di base .....	41
Nozioni di base .....	67
Azioni .....	44
IAM .....	179
Nozioni di base .....	41
Nozioni di base .....	67
Azioni .....	44
Kinesis .....	236
Esempi serverless .....	64
AWS KMS .....	238
Azioni .....	44
Lambda .....	242
Nozioni di base .....	41
Nozioni di base .....	67
Azioni .....	44
Scenari .....	63
Esempi serverless .....	64
MSK Amazon .....	272
Esempi serverless .....	64

Amazon Polly .....	273
Azioni .....	44
Scenari .....	63
Amazon RDS .....	277
Nozioni di base .....	41
Azioni .....	44
Esempi serverless .....	64
Simple Storage Service (Amazon S3) .....	285
Nozioni di base .....	41
Nozioni di base .....	67
Azioni .....	44
Scenari .....	63
Esempi serverless .....	64
Amazon SES .....	317
Azioni .....	44
API Amazon SES v2 .....	323
Azioni .....	44
Amazon SNS .....	324
Azioni .....	44
Esempi serverless .....	64
Amazon SQS .....	334
Azioni .....	44
Esempi serverless .....	64
AWS STS .....	347
Azioni .....	44
Amazon Textract .....	349
Scenari .....	63
Amazon Translate .....	350
Scenari .....	63
Migrazione delle versioni .....	352
Side-by-side utilizzo .....	352
Differenze generali .....	352
Differenze tra i client .....	353
differenze di risorse .....	354
Sicurezza .....	356
Protezione dei dati .....	356

---

Identity and Access Management .....	357
Convalida della conformità .....	358
Resilienza .....	359
Sicurezza dell'infrastruttura .....	359
Applicazione di una versione minima di TLS .....	360
Verifica della versione OpenSSL .....	360
Aggiornamento del supporto TLS .....	361
Migrazione del client di crittografia S3 (da V1 a V2) .....	361
Panoramica sulla migrazione .....	361
Aggiorna i client esistenti per leggere nuovi formati .....	361
Esegui la migrazione dei client di crittografia e decrittografia alla versione 2 .....	363
Migrazione del client di crittografia S3 (da V2 a V3) .....	366
Panoramica sulla migrazione .....	366
Comprensione delle funzionalità della V3 .....	367
Aggiorna i client esistenti per leggere nuovi formati .....	370
Migrazione dei client di crittografia e decrittografia alla V3 .....	371
Cronologia dei documenti .....	379
.....	ccclxxxi

# Cos'è l' AWS SDK for Ruby?

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

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

[Guida introduttiva all' AWS SDK for Ruby](#)

## Documentazione e risorse aggiuntive

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

- [AWS SDKs e guida di riferimento agli strumenti](#): contiene impostazioni, funzionalità e altri concetti fondamentali comuni a AWS SDKs
- [AWS SDK per Ruby Riferimento API - Versione 3](#)
- [AWS Repository di esempi di codice](#) su GitHub
- [RubyGems.org](#) — L'ultima versione di SDK è modularizzata in gemme specifiche per i servizi disponibili qui
  - [Servizi supportati](#): elenca tutte le gemme supportate dall' AWS SDK for Ruby
- AWS Sorgente SDK for Ruby GitHub su:
  - [Fonte e README](#)
  - [Registri delle modifiche sotto ogni gemma](#)
  - [Passaggio dalla v2 alla v3](#)
  - [Problemi](#)
  - [Note di aggiornamento principali](#)
- [Blog per sviluppatori](#)

## Implementazione nel cloud AWS

È possibile utilizzare Servizi AWS tali applicazioni AWS Elastic Beanstalk e AWS CodeDeploy distribuirle nel AWS cloud. Per la distribuzione di applicazioni Ruby con Elastic Beanstalk, [consulta](#)

[Deploying Elastic Beanstalk Applications in Ruby Using EB CLI e Git nella Developer Guide](#). [AWS Elastic Beanstalk](#) [Per una panoramica dei servizi di distribuzione, vedi Panoramica delle opzioni di distribuzione su. AWSAWS](#)

## Manutenzione e supporto per le versioni principali dell'SDK

Per informazioni sulla manutenzione e il supporto per le versioni principali dell'SDK e le relative dipendenze sottostanti, consulta quanto segue nella Guida di [riferimento agli strumenti AWS SDKs e agli strumenti](#):

- [AWS SDKs e politica di manutenzione degli strumenti](#)
- [AWS SDKs e Tools Version Support Matrix](#)

# Guida introduttiva all' AWS SDK for Ruby

Scopri come installare, configurare e utilizzare l'SDK per creare un'applicazione Ruby per accedere a una AWS risorsa a livello di codice.

## Argomenti

- [Autenticazione con l' AWS utilizzo di AWS SDK for Ruby](#)
- [Installazione dell' AWS SDK per Ruby](#)
- [Creazione di una semplice applicazione utilizzando l' AWS SDK for Ruby](#)

## Autenticazione con l' AWS utilizzo di AWS SDK for Ruby

È necessario stabilire in che modo il codice si autentica durante lo sviluppo con AWS . Servizi AWS È possibile configurare l'accesso programmatico alle AWS risorse in diversi modi a seconda dell'ambiente e dell' AWS accesso a disposizione.

Per scegliere il metodo di autenticazione e configurarlo per l'SDK, consulta [Autenticazione e accesso](#) nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

## Utilizzo delle credenziali della console

Per lo sviluppo locale, consigliamo ai nuovi utenti di utilizzare le credenziali di accesso esistenti alla Console di AWS gestione per l'accesso programmatico ai servizi. AWS Dopo l'autenticazione basata su browser, AWS genera credenziali temporanee che funzionano con strumenti di sviluppo locali come l'interfaccia a riga di AWS comando (AWS CLI) e l'SDK for Ruby. AWS

Se scegli questo metodo, segui le istruzioni [per accedere allo sviluppo AWS locale utilizzando le credenziali della console utilizzando la AWS CLI](#).

L' AWS SDK for Ruby non necessita di gemme aggiuntive (`aws-sdk-signinad` esempio) da aggiungere all'applicazione per utilizzare l'accesso con le credenziali della console.

## Utilizzo dell'autenticazione IAM Identity Center

Se scegli questo metodo, completa la procedura per l'[autenticazione IAM Identity Center](#) nella AWS SDKs and Tools Reference Guide. Successivamente, l'ambiente dovrebbe contenere i seguenti elementi:

- Il AWS CLI, che viene utilizzato per avviare una sessione del portale di AWS accesso prima di eseguire l'applicazione.
- Un [AWSconfigfile 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 agli strumenti AWS SDKs e strumenti.
- Il config file condiviso imposta l'[region](#) impostazione. Questo imposta l'impostazione predefinita Regione AWS utilizzata dall'SDK per AWS le richieste. Questa regione viene utilizzata per le richieste di servizio SDK che non sono specificate con una regione da utilizzare.
- L'SDK utilizza la [configurazione del provider di token SSO](#) del profilo per acquisire le credenziali prima di inviare richieste a. AWS Il sso\_role\_name valore, che è un ruolo IAM connesso a un set di autorizzazioni IAM Identity Center, consente l'accesso ai dati Servizi AWS utilizzati 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-sessione](#) denominata. La sso-session sezione contiene le impostazioni per avviare una sessione del portale di AWS 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' AWS SDK for Ruby non necessita di gemme aggiuntive (aws-sdk-ssoaws-sdk-ssooidc come) da aggiungere all'applicazione per utilizzare l'autenticazione IAM Identity Center.

## Avvia una sessione del portale di accesso AWS

Prima di eseguire un'applicazione che consente l'accesso Servizi AWS, è necessaria una sessione attiva del portale di AWS accesso affinché l'SDK utilizzi l'autenticazione IAM Identity Center per risolvere le credenziali. A seconda della durata della sessione configurata, l'accesso alla fine scadrà

e l'SDK riscontrerà un errore di autenticazione. Per accedere al portale di AWS accesso, esegui il seguente comando in. AWS CLI

```
aws sso login
```

Se hai seguito le istruzioni e disponi di una configurazione predefinita del profilo, non è necessario richiamare 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 il AWS 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 di AWS accesso ed esegui `aws sso login`, non ti verrà richiesto di fornire credenziali.

La procedura di accesso potrebbe richiedere all'utente di consentire l' AWS CLI accesso ai dati. Poiché AWS CLI è basato sull'SDK per Python, i messaggi di autorizzazione potrebbero contenere variazioni del `botocore` nome.

## Ulteriori informazioni di 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 agli AWS ambienti e alle applicazioni dell'utente. Gli utenti umani che fanno parte della tua organizzazione, ovvero tu, lo sviluppatore, sono noti come identità della forza lavoro.

Utilizza credenziali temporanee per l'accesso. AWS Puoi utilizzare un provider di identità per i tuoi utenti umani per fornire l'accesso federato agli AWS account assumendo ruoli che forniscono credenziali temporanee. Per la gestione centralizzata degli accessi, ti consigliamo di utilizzare AWS IAM Identity Center (IAM Identity Center) per gestire l'accesso ai tuoi account e le autorizzazioni all'interno di tali account. Per altre alternative, consulta quanto segue:

- Per ulteriori informazioni sulle best practice, consulta [Best practice per la sicurezza in IAM](#) nella Guida per l'utente di IAM.
- Per creare AWS credenziali a breve termine, consulta [Temporary Security Credentials](#) nella IAM User Guide.
- Per ulteriori informazioni sulla catena di provider di credenziali AWS SDK for Ruby e su come i diversi metodi di autenticazione vengono tentati automaticamente dall'SDK in una sequenza, consulta. [Catena di fornitori di credenziali](#)
- Per le impostazioni di configurazione dei provider di credenziali AWS SDK, consulta [Fornitori di credenziali standardizzati](#) nella and Tools Reference Guide.AWS SDKs

## Installazione dell' AWS SDK per Ruby

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

### Prerequisiti

Prima di utilizzare l' AWS SDK for Ruby, è necessario autenticarsi con. AWS Per informazioni sulla configurazione dell'autenticazione, consulta. [Autenticazione con l' AWS utilizzo di AWS SDK for Ruby](#)

### Installazione dell'SDK

Puoi installare l' AWS SDK per Ruby come faresti con qualsiasi altra gemma Ruby. Le gemme sono disponibili all'indirizzo. [RubyGems](#) L' AWS SDK for Ruby è progettato per essere modulare ed è separato da. Servizio AWS L'installazione dell'intera aws - sdk gemma è complessa e può richiedere più di un'ora.

Ti consigliamo di installare solo le gemme 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 AWS file README SDK for Ruby. Ad esempio, la gemma 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 utilizzi un sistema operativo Amazon Linux 2, puoi usare i seguenti comandi per aggiornare RVM, elencare le versioni di Ruby disponibili, quindi scegliere la versione che desideri utilizzare per lo sviluppo con l'SDK for AWS Ruby. La versione minima richiesta di Ruby è 2.5.

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

## Bundler

Se usi [Bundler](#), i seguenti comandi installano la gem AWS SDK for Ruby 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 utilizzerà il tuo codice. Per seguire l'esempio di Amazon S3, aggiungi 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
```

## Creazione di una semplice applicazione utilizzando l' AWS SDK for Ruby

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

## Scrivere 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__
```

AWS SDK for Ruby è progettato per essere modulare ed è separato da. Servizio AWS Dopo l'installazione della gem, l'installazione 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 tuo 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`

## Fasi successive

Per testare molte altre operazioni di Amazon S3, consulta il [AWS Code Examples Repository](#) su GitHub

# Configurazione dei client di servizio nell' AWS SDK for Ruby

Per accedere a livello di codice Servizi AWS, l' AWS SDK for Ruby utilizza una classe client per ciascuno. Servizio AWS Ad esempio, se la tua applicazione deve accedere ad Amazon EC2, crea un oggetto EC2 client Amazon per interfacciarsi con quel servizio. Quindi utilizzi il client del servizio per effettuare richieste in merito Servizio AWS.

Per fare una richiesta a un Servizio AWS, devi prima creare un client di servizio. Per ogni Servizio AWS utilizzo del codice, ha la propria gemma e il proprio tipo dedicato per interagire con essa. Il client espone un metodo per ogni operazione API esposta dal servizio.

Esistono molti modi alternativi per configurare il comportamento dell'SDK, ma alla fine tutto ha a che fare con il comportamento dei client di servizio. Qualsiasi configurazione non ha effetto finché non viene utilizzato un client di servizio creato a partire da tali configurazioni.

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.

La [AWS SDKs and Tools Reference Guide](#) contiene anche impostazioni, funzionalità e altri concetti fondamentali comuni a molti di AWS SDKs.

## Argomenti

- [Precedenza delle impostazioni](#)
- [Configurazione esterna dell' AWS SDK per i client del servizio Ruby](#)
- [Configurazione dell' AWS SDK per i client del servizio Ruby nel codice](#)
- [Impostazione dell' Regione AWS AWS SDK for Ruby](#)
- [Utilizzo di AWS SDK for Ruby per i provider di credenziali](#)
- [Configurazione dei nuovi tentativi nell' AWS SDK per Ruby](#)
- [Configurazione delle funzionalità di osservabilità nell' AWS SDK for Ruby](#)
- [Configurazione delle impostazioni a livello HTTP all'interno dell' AWS SDK for Ruby](#)

È possibile utilizzare [configShared and credentials Files](#) per le impostazioni di configurazione. Per tutte le impostazioni AWS SDK, consulta il [riferimento alle impostazioni nella Guida di riferimento](#) agli strumenti AWS SDKs e agli strumenti.

È possibile utilizzare profili diversi per memorizzare configurazioni diverse. Per specificare il profilo attivo caricato dall'SDK, puoi utilizzare la variabile di `AWS_PROFILE` ambiente o l'`profile` opzione di `Aws.config`

## Precedenza delle impostazioni

Le impostazioni globali configurano funzionalità, fornitori di credenziali e altre funzionalità che sono supportate dalla maggior parte SDKs e hanno un ampio impatto su tutti. Servizi AWS Tutte AWS SDKs hanno una serie di luoghi (o fonti) che controllano per trovare un valore per le impostazioni globali. Non tutte le impostazioni sono disponibili in tutte le fonti. Di seguito è riportata la priorità di impostazione della ricerca:

1. Qualsiasi impostazione esplicita impostata nel codice o su un client di servizio stesso ha la precedenza su qualsiasi altra cosa.
  - a. Tutti i parametri passati direttamente a un costruttore del client hanno la massima priorità.
  - b. `Aws.config` viene controllato per le impostazioni globali o specifiche del servizio.
2. La variabile di ambiente è selezionata.
3. Il `AWS credentials` file condiviso viene controllato.
4. Il `AWS config` file condiviso viene controllato.
5. Qualsiasi valore predefinito fornito dallo stesso codice sorgente AWS SDK for Ruby viene utilizzato per ultimo.

## Configurazione esterna dell' AWS SDK per i client del servizio Ruby

Molte impostazioni di configurazione possono essere gestite al di fuori del codice. Quando la configurazione viene gestita esternamente, viene applicata a tutte le applicazioni. La maggior parte delle impostazioni di configurazione può essere impostata come variabili di ambiente o in un file condiviso separato. `AWS config` Il `config` file condiviso può mantenere set di impostazioni separati, chiamati profili, per fornire configurazioni diverse per ambienti o test diversi.

Le variabili di ambiente e le impostazioni dei `config` file condivisi sono standardizzate e condivise tra AWS SDKs strumenti per supportare funzionalità coerenti tra diversi linguaggi di programmazione e applicazioni.

Consulta la [AWS SDKs and Tools Reference Guide](#) per ulteriori informazioni sulla configurazione dell'applicazione con questi metodi, oltre a dettagli su ciascuna impostazione cross-sdk. Per

visualizzare tutte le impostazioni che l'SDK è in grado di risolvere a partire dalle variabili di ambiente o dai file di configurazione, consulta il [riferimento alle impostazioni nella Guida di riferimento](#) agli strumenti AWS SDKs e agli strumenti.

Per effettuare una richiesta a un Servizio AWS, devi prima creare un'istanza di un client per quel servizio. È possibile configurare impostazioni comuni per i client di servizio, ad esempio i timeout, il client HTTP e riprovare la configurazione.

Ogni client di servizio richiede un fornitore di credenziali Regione AWS e un provider di credenziali. L'SDK utilizza questi valori per inviare le richieste alla regione corretta per le tue risorse e per firmare le richieste con le credenziali corrette. Puoi specificare questi valori a livello di codice a livello di codice o caricarli automaticamente dall'ambiente.

L'SDK ha una serie di posizioni (o fonti) che controlla per trovare un valore per le impostazioni di configurazione.

1. Qualsiasi impostazione esplicita impostata nel codice o su un client di servizio stesso ha la precedenza su qualsiasi altra cosa.
2. Variabili di ambiente
  - Per i dettagli sull'impostazione delle variabili di ambiente, consultate le [variabili di ambiente nella Guida](#) di riferimento agli strumenti AWS SDKs e agli strumenti.
  - Nota che puoi configurare le variabili di ambiente per una shell a diversi livelli di ambito: a livello di sistema, a livello di utente e per una sessione di terminale specifica.
3. `Condivisi` e file `config credentials`
  - Per i dettagli sulla configurazione di questi file, consulta la Guida di riferimento [Condivisi configAWS SDKs e credentials file](#) in and Tools.
4. Qualsiasi valore predefinito fornito dal codice sorgente SDK stesso viene utilizzato per ultimo.
  - Alcune proprietà, come `Region`, non hanno un valore predefinito. È necessario specificarle esplicitamente nel codice, in un'impostazione di ambiente o nel `config` file condiviso. Se l'SDK non è in grado di risolvere la configurazione richiesta, le richieste API possono avere esito negativo in fase di esecuzione.

## AWS SDK per variabili di ambiente Ruby

Oltre alle [variabili di ambiente cross-sdk](#) supportate dalla maggior parte AWS SDKs, l' AWS SDK for Ruby ne supporta alcune uniche:

## AWS\_SDK\_CONFIG\_OPT\_OUT

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

## AMAZON\_REGION

Una variabile di ambiente alternativa a quella `AWS_REGION` per l'impostazione di Regione AWS. Questo valore viene controllato solo se non `AWS_REGION` viene utilizzato.

# Configurazione dell' AWS SDK per i client del servizio Ruby nel codice

Quando la configurazione viene gestita direttamente nel codice, l'ambito della configurazione è limitato all'applicazione che utilizza quel codice. All'interno di tale applicazione, sono disponibili opzioni per la configurazione globale di tutti i client di servizio, la configurazione per tutti i client di un determinato Servizio AWS tipo o la configurazione per un'istanza specifica del client di servizio.

## Aws.config

Per fornire una configurazione globale all'interno del codice per tutte le AWS classi, utilizzate [Aws.config](#) quella disponibile nella `aws-sdk-core` gem.

`Aws.config` supporta due sintassi per usi diversi. Le impostazioni globali possono essere applicate a tutti i Servizi AWS o a un servizio specifico. Per l'elenco completo delle impostazioni supportate, consulta l'[ClientOptions AWS SDK per Ruby API Reference](#).

## Impostazioni globali tramite **Aws.config**

Per configurare impostazioni indipendenti dal servizio `Aws.config`, utilizzate la seguente sintassi:

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

Queste impostazioni vengono unite a tutti i client di servizio creati.

Esempio di impostazione globale:

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

Se si tenta di utilizzare un nome di impostazione che non è supportato a livello globale, viene generato un errore quando si tenta di creare un'istanza di un tipo di servizio che non lo supporta. In tal caso, utilizza invece la sintassi specifica del servizio.

## Impostazioni specifiche del servizio tramite `Aws.config`

Per configurare impostazioni specifiche del servizio `Aws.config`, utilizza la seguente sintassi:

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

Queste impostazioni vengono unite a tutti i client di servizio creati per quel tipo di servizio.

Esempio di impostazione che si applica solo ad Amazon S3:

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

`<service identifier>` È possibile identificarli osservando il nome della gem [AWS SDK for Ruby](#) corrispondente e utilizzando il suffisso che segue "». `aws-sdk` - Esempio:

- Infatti `aws-sdk-s3`, la stringa identificativa del servizio è "». `s3`
- Infatti `aws-sdk-ecs`, la stringa dell'identificatore del servizio è "»`ecs`.

## Impostazione dell' Regione AWS AWS SDK for Ruby

È possibile accedere a Servizi AWS ciò che opera in un'area geografica specifica utilizzando. Regioni AWS Ciò può essere utile sia per la ridondanza sia per mantenere attivi i dati e le applicazioni vicino a dove voi e i vostri utenti vi accedete.

### Important

La maggior parte delle risorse risiede in una regione specifica Regione AWS ed è necessario fornire la regione corretta per la risorsa quando si utilizza l'SDK.

È necessario impostare un valore predefinito Regione AWS per l'SDK for Ruby da utilizzare per le richieste. AWS Questa impostazione predefinita viene utilizzata per tutte le chiamate ai metodi di servizio SDK che non sono specificate con una regione.

Per ulteriori informazioni sull'impostazione della regione, consulta la Guida [Regione AWS](#) di riferimento agli strumenti AWS SDKs e agli strumenti. Sono inclusi anche esempi su come impostare l'area predefinita tramite il `AWS config` file condiviso o le variabili di ambiente.

## Ordine di ricerca regionale per la risoluzione

È necessario impostare una regione quando si utilizza la maggior parte Servizi AWS. L'AWS SDK 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 le variabili di ambiente
4. Impostazione della regione utilizzando il `config` file condiviso

## Come impostare la regione

Questa sezione descrive diversi modi per 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 AWS SDKs and Tools Reference Guide](#).

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

```
[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 client o in un oggetto 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')
```

## Utilizzo di AWS SDK for Ruby per i provider di credenziali

Tutte le richieste AWS devono essere firmate crittograficamente utilizzando credenziali emesse da AWS. In fase di esecuzione, l'SDK recupera i valori di configurazione delle credenziali controllando diverse posizioni.

L'autenticazione con AWS può essere gestita all'esterno del codebase. Molti metodi di autenticazione possono essere rilevati, utilizzati e aggiornati automaticamente dall'SDK utilizzando la catena di fornitori di credenziali.

Per le opzioni guidate su come iniziare a utilizzare AWS l'autenticazione del progetto, consultate [Autenticazione e accesso](#) nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

## Catena di fornitori di credenziali

Tutti SDKs hanno una serie di luoghi (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 chiamata catena di fornitori di credenziali predefinita.

### Note

Se hai seguito l'approccio consigliato per i nuovi utenti per iniziare, ti sei autenticato utilizzando l'accesso con le credenziali della console durante. [Autenticazione con l' AWS utilizzo di AWS SDK for Ruby](#) 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 strumenti AWS SDKs e agli strumenti.

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.

La AWS SDKs and Tools Reference Guide contiene informazioni sulle impostazioni di configurazione SDK utilizzate da tutti AWS SDKs 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'autenticazione AWS, l' AWS SDK for Ruby controlla i provider di credenziali nell'ordine elencato nella tabella seguente.

Fornitore di credenziali per precedenza	AWS SDKs e guida di riferimento agli strumenti	AWS SDK per Ruby Documentazione di riferimento delle API
AWS chiavi di accesso (credenziali temporanee e a lungo termine)	<a href="#">AWS chiavi di accesso</a>	<a href="#">Aws::Credentials</a> <a href="#">Aws::SharedCredentials</a>

Fornitore di credenziali per precedenza	AWS SDKs e guida di riferimento agli strumenti	AWS SDK per Ruby Documentazione di riferimento delle API
Token di identità Web da AWS Security Token Service (AWS STS)	<a href="#">Assumi il ruolo di fornitore di credenziali</a>  Utilizzando <code>role_arn</code> , <code>role_session_name</code> , e <code>web_identity_token_file</code>	<a href="#">Aws::AssumeRoleWebIdentityCredentials</a>
AWS IAM Identity Center. In questa guida, vedi <a href="#">Autenticazione con l' AWS utilizzo di AWS SDK for Ruby</a> .	<a href="#">Provider di credenziali IAM Identity Center</a>	<a href="#">Aws::SSOCredentials</a>
Fornitore di entità affidabile (ad esempio <code>AWS_ROLE_ARN</code> ). In questa guida, vedi <a href="#">Creazione di un token di accesso AWS STS</a> .	<a href="#">Assumi il ruolo di fornitore di credenziali</a>  Utilizzando <code>role_arn</code> e <code>role_session_name</code>	<a href="#">Aws::AssumeRoleCredentials</a>
Fornitore di credenziali di accesso	<a href="#">Fornitore di credenziali di accesso</a>	<a href="#">Aws::LoginCredentials</a>
Fornitore di credenziali di processo	<a href="#">Fornitore di credenziali di processo</a>	<a href="#">Aws::ProcessCredentials</a>
Credenziali Amazon Elastic Container Service (Amazon ECS)	<a href="#">Fornitore di credenziali per container</a>	<a href="#">Aws::ECSCredentials</a>
Credenziali del profilo di istanza Amazon Elastic Compute Cloud (Amazon EC2) (provider di credenziali IMDS)	<a href="#">Fornitore di credenziali IMDS</a>	<a href="#">Aws::InstanceProfileCredentials</a>

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

## Creazione di un token di accesso AWS STS

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 `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 AWS SDKs and Tools Reference Guide.

## Configurazione dei nuovi tentativi nell' AWS SDK per Ruby

L' AWS SDK for Ruby offre un comportamento di riprova predefinito per le richieste di servizio e opzioni di configurazione personalizzabili. Chiamate per restituire Servizi AWS occasionalmente eccezioni impreviste. Alcuni tipi di errori, come gli errori di limitazione o gli errori transitori, potrebbero avere esito positivo se la chiamata viene ritentata.

Il comportamento dei tentativi può essere configurato globalmente utilizzando le variabili o le impostazioni di ambiente nel file condiviso. AWS config Per informazioni su questo approccio, consulta [Retry behavior nella AWS SDKs and Tools Reference](#) Guide. Include anche informazioni dettagliate sulle implementazioni della strategia Retry e su come sceglierne una piuttosto che un'altra.

In alternativa, queste opzioni possono essere configurate anche nel codice, come illustrato nelle sezioni seguenti.

## Specificazione del comportamento di ripetizione dei tentativi del client nel codice

Per impostazione predefinita, l' AWS SDK 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) }  
)
```

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`

## Configurazione delle funzionalità di osservabilità nell' AWS SDK for Ruby

L'osservabilità è la misura in cui lo stato attuale di un sistema può essere dedotto dai dati che emette. I dati emessi vengono comunemente definiti telemetria. L' AWS SDK for Ruby può fornire le tracce come segnale di telemetria. È possibile collegare un file `TelemetryProvider` per raccogliere e inviare dati di telemetria a un backend di osservabilità. [L'SDK attualmente supporta OpenTelemetry \(OTel\) come provider di telemetria e OpenTelemetry offre molti modi per esportare i dati di telemetria, incluso l'utilizzo di Amazon. AWS X-Ray CloudWatch Per ulteriori informazioni sugli esportatori di Ruby, OpenTelemetry consulta Esportatori sul sito Web.](#) `OpenTelemetry`

Per impostazione predefinita, l'SDK non registra né emette dati di telemetria. Questo argomento spiega come configurare ed emettere l'output di telemetria.

La telemetria può essere configurata per un servizio specifico o a livello globale. L'SDK for Ruby fornisce un provider OpenTelemetry . Puoi anche definire un provider di telemetria personalizzato a tua scelta.

## Configurazione di un client per un servizio **OTelProvider**

L'SDK for Ruby OpenTelemetry fornisce un provider chiamato. [OTelProvider](#) L'esempio seguente configura l'esportazione di telemetria utilizzando per OpenTelemetry il client del servizio Amazon Simple Storage Service. Per questo semplice esempio, la variabile di OTEL\_TRACES\_EXPORTER ambiente from OpenTelemetry viene utilizzata per esportare le tracce nell'output della console quando esegui il codice. Per ulteriori informazioni OTEL\_TRACES\_EXPORTER, consulta [Exporter Selection](#) nella OpenTelemetry documentazione.

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

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

OpenTelemetry::SDK.configure

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

L'esempio di codice precedente mostra i passaggi per configurare l'output di traccia per un client di servizio:

1. Richiedi OpenTelemetry dipendenze.
  - a. [opentelemetry-sdk](#) per l'utilizzo `Aws::Telemetry::OTelProvider`.
  - b. [opentelemetry-exporter-otlp](#) per esportare dati di telemetria.
2. Chiama `OpenTelemetry::SDK.configure` per configurare l' OpenTelemetry SDK con le impostazioni di configurazione predefinite.
3. Usando SDK per il OpenTelemetry provider di Ruby, crea un'istanza di `OTelProvider` passare come opzione di configurazione al client di servizio che desideri tracciare.

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

Utilizzando questi passaggi, tutti i metodi richiamati su quel client di servizio emetteranno dati di traccia.

Un esempio dell'output di traccia generato dalla chiamata al `list_buckets` metodo di Amazon S3 è il seguente:

### Esempio di output di OpenTelemetry traccia

```
#<struct OpenTelemetry::SDK::Trace::SpanData
  name="Handler.NetHttp",
  kind=:internal,
  status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
  parent_span_id="\xBFb\xC9\xFD\xA6F!\xE1",
  total_recorded_attributes=7,
  total_recorded_events=0,
  total_recorded_links=0,
  start_timestamp=1736190567061767000,
  end_timestamp=1736190567317160000,
  attributes=
    {"http.method"=>"GET",
     "net.protocol.name"=>"http",
     "net.protocol.version"=>"1.1",
     "net.peer.name"=>"s3.amazonaws.com",
     "net.peer.port"=>"443",
     "http.status_code"=>"200",
     "aws.request_id"=>"22HSH7NQTYMB5NHQ"},
  links=nil,
  events=nil,
  resource=
    #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
      @attributes=
        {"service.name"=>"unknown_service",
         "process.pid"=>37013,
         "process.command"=>"example.rb",
         "process.runtime.name"=>"ruby",
         "process.runtime.version"=>"3.3.0",
         "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
[arm64-darwin23]"},
         "telemetry.sdk.name"=>"opentelemetry",
         "telemetry.sdk.language"=>"ruby",
         "telemetry.sdk.version"=>"1.6.0"}>,
  instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
    name="aws.s3.client", version="">
```

```

span_id="\xEF%\x9C\xB5\x8C\x04\xDB\x7F",
trace_id=" \xE7\xF1\xF8\x9D\xe\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
#<struct OpenTelemetry::SDK::Trace::SpanData
name="S3.ListBuckets",
kind=:client,
status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
parent_span_id="\x00\x00\x00\x00\x00\x00\x00",
total_recorded_attributes=5,
total_recorded_events=0,
total_recorded_links=0,
start_timestamp=1736190567054410000,
end_timestamp=1736190567327916000,
attributes={"rpc.system"=>"aws-api", "rpc.service"=>"S3", "rpc.method"=>"ListBuckets",
"code.function"=>"list_buckets", "code.namespace"=>"Aws::Plugins::Telemetry"},
links=nil,
events=nil,
resource=
#<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
@attributes=
{"service.name"=>"unknown_service",
"process.pid"=>37013,
"process.command"=>"example.rb",
"process.runtime.name"=>"ruby",
"process.runtime.version"=>"3.3.0",
"process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
[arm64-darwin23]"},
"telemetry.sdk.name"=>"opentelemetry",
"telemetry.sdk.language"=>"ruby",
"telemetry.sdk.version"=>"1.6.0"}>,
instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
name="aws.s3.client", version="">,
span_id="\xBFb\xC9\xFD\xA6F!\xE1",
trace_id=" \xE7\xF1\xF8\x9D\xe\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>

```

L'output di traccia precedente ha due intervalli di dati. Ogni voce di traccia fornisce metadati aggiuntivi sull'evento in uno o più attributi.

## Configurazione di un servizio **OTelProvider** per tutti i client di servizio

Invece di attivare la telemetria per un client di servizio specifico come spiegato nella sezione precedente, è possibile attivare la telemetria a livello globale.

Per emettere dati di telemetria per tutti i client di AWS servizio, puoi impostare il provider di telemetria prima di creare i client di servizio. `Aws.config`

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

Con questa configurazione, tutti i client di servizio creati successivamente emetteranno automaticamente dati di telemetria. Per ulteriori informazioni sull'utilizzo per configurare le impostazioni globali, `Aws.config` consulta [Aws.config](#)

## Configurazione di un provider di telemetria personalizzato

Se non desideri utilizzarlo OpenTelemetry come provider di telemetria, l'SDK for AWS Ruby ti supporta nell'implementazione di un provider personalizzato. Potrebbe essere utile utilizzare come esempio l'[OTelProviderimplementazione](#) disponibile nel repository AWS SDK GitHub for Ruby. Per un contesto aggiuntivo, consulta le note contenute [Module: Aws::Telemetry](#) nell'API Reference.AWS SDK per Ruby

## Attributi Span

Le tracce sono l'output della telemetria. Le tracce sono costituite da uno o più intervalli. Gli span hanno attributi che includono metadati aggiuntivi che vengono inclusi automaticamente quando appropriato per la chiamata al metodo. Di seguito è riportato un elenco degli attributi supportati dall'SDK for Ruby, in cui:

- Nome dell'attributo: il nome usato per etichettare i dati che appaiono nella traccia.
- Tipo: il tipo di dati del valore.
- Descrizione: una descrizione di ciò che rappresenta il valore.

Nome attributo	Tipo	Descrizione
----------------	------	-------------

<code>error</code>	Booleano	Vero se l'unità di lavoro non ha avuto successo. In caso contrario, false.
<code>exception.message</code>	Stringa	L'eccezione o il messaggio di errore.
<code>exception.stacktrace</code>	Stringa	Uno stacktrace fornito dal runtime del linguaggio, se disponibile.
<code>exception.type</code>	Stringa	Il tipo (nome completo) dell'eccezione o dell'errore.
<code>rpc.system</code>	Stringa	L'identificatore di sistema remoto impostato su 'aws-api'.
<code>rpc.method</code>	Stringa	Il nome dell'operazione che viene richiamata.
<code>rpc.service</code>	Stringa	Il nome del servizio remoto.
<code>aws.request_id</code>	Stringa	L'ID della AWS richiesta restituito nelle intestazioni di risposta, per tentativo HTTP. L'ID della richiesta più recente viene utilizzato quando possibile.
<code>code.function</code>	Stringa	Il nome del metodo o della funzione.
<code>code.namespace</code>	Stringa	Lo spazio dei nomi all'interno del quale <code>code.function</code> è definito.
<code>http.status_code</code>	Long	Il codice di stato della risposta HTTP.

<code>http.request_content_length</code>	Long	La dimensione del corpo del payload della richiesta in byte.
<code>http.response_content_length</code>	Long	La dimensione del corpo del payload della risposta in byte.
<code>http.method</code>	Stringa	Il metodo della richiesta HTTP.
<code>net.protocol.name</code>	Stringa	Il nome del protocollo a livello di applicazione.
<code>net.protocol.version</code>	Stringa	La versione del protocollo a livello applicativo (ad esempio 1.0, 1.1, 2.0).
<code>net.peer.name</code>	Stringa	Il nome host remoto logico.
<code>net.peer.port</code>	Stringa	Il numero di porta remota logica.

### Tip

OpenTelemetry-Ruby ha implementazioni aggiuntive integrate con SDK per il supporto di telemetria esistente di Ruby. [Per ulteriori informazioni, consulta -SDK Instrumentation nel repository. OpenTelemetry AWS open-telemetry GitHub](#)

## Configurazione delle impostazioni a livello HTTP all'interno dell'AWS SDK for Ruby

### 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`.

```
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 persista, consulta l'opzione di configurazione degli [endpoint specifici del servizio](#) nella and Tools Reference Guide.AWS SDKs

# Usare l' AWS SDK per Ruby

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

La [AWS SDKs and Tools Reference Guide](#) contiene anche impostazioni, funzionalità e altri concetti fondamentali comuni a molti di. AWS SDKs

## Argomenti

- [Effettuare Servizio AWS richieste utilizzando l' AWS SDK for Ruby](#)
- [Utilizzo dell'utilità AWS SDK for Ruby REPL](#)
- [Usare l' AWS SDK per Ruby con Ruby on Rails](#)
- [Eseguire il debug utilizzando le informazioni di wire trace da un client SDK for AWS Ruby](#)
- [Aggiungere test con stubbing all'applicazione AWS SDK for Ruby](#)
- [Utilizzo dei risultati impaginati nell' AWS SDK for Ruby](#)
- [Usare i camerieri nell' AWS SDK for Ruby](#)

## Effettuare Servizio AWS richieste utilizzando l' AWS SDK for Ruby

Per accedere a livello di codice Servizi AWS, SDKs usa una classe client per ciascuno. Servizio AWS Ad esempio, se la tua applicazione deve accedere ad Amazon EC2, crea un oggetto EC2 client Amazon per interfacciarsi con quel servizio. Quindi utilizzi il client del servizio per effettuare richieste in merito Servizio AWS.

Per fare una richiesta a un Servizio AWS, devi prima creare e [configurare](#) un client di servizio. Per ogni Servizio AWS codice utilizzato, ha la propria gemma e il proprio tipo dedicato per interagire con esso. Il client espone un metodo per ogni operazione API esposta dal servizio.

Ogni client di servizio richiede un fornitore di credenziali Regione AWS e un provider di credenziali. L'SDK utilizza questi valori per inviare le richieste alla regione corretta per le tue risorse e per firmare le richieste con le credenziali corrette. Puoi specificare questi valori a livello di codice a livello di codice o caricarli automaticamente dall'ambiente.

- Quando si crea un'istanza di una classe client, è necessario fornire AWS le credenziali. Per l'ordine in cui l'SDK verifica i provider di autenticazione, consulta. [Catena di fornitori di credenziali](#)

- L'SDK ha una serie di posizioni (o fonti) che controlla per trovare un valore per le impostazioni di configurazione. Per informazioni dettagliate, consultare [Precedenza delle impostazioni](#).

L'SDK for Ruby include classi client che forniscono interfacce a Servizi AWS. Ogni classe client supporta una particolare classe Servizio AWS e segue la convenzione. `Aws::<service identifier>::Client`. Ad esempio, `Aws::S3::Client` fornisce un'interfaccia per il servizio Amazon Simple Storage Service e `Aws::SQS::Client` fornisce un'interfaccia per il servizio Amazon Simple Queue Service.

Tutte le classi di client Servizi AWS sono thread-safe per tutti.

È possibile passare le opzioni di configurazione direttamente ai costruttori Client e Resource. Queste opzioni hanno la precedenza sull'ambiente e `Aws.config` sui valori predefiniti.

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

## Utilizzo dell'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

- [Installazione dell' AWS SDK per 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' AWS SDK 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 minuscolo. Ad esempio, i nomi degli oggetti EC2 helper di Amazon S3 e Amazon sono rispettivamente `s3` e `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.

## Usare l' AWS SDK per Ruby con Ruby on Rails

[Ruby on Rails](#) fornisce un framework di sviluppo web che semplifica la creazione di siti Web con Ruby.

AWS fornisce la `aws-sdk-rails` gemma per consentire una facile integrazione con Rails. Puoi usare AWS Elastic Beanstalk AWS OpsWorks AWS CodeDeploy, o [AWS Rails Provisioner](#) per distribuire ed eseguire le tue applicazioni Rails nel cloud. AWS

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

## Eseguire il debug utilizzando le informazioni di wire trace da un client SDK for AWS Ruby

È possibile ottenere informazioni sul wire trace da un AWS client impostando il valore `http_wire_trace` booleano. Le informazioni sul tracciamento elettronico aiutano a distinguere le modifiche dei client, i problemi di servizio e gli errori degli utenti. Quando `true`, l'impostazione mostra ciò che viene inviato sul cavo. L'esempio seguente crea un client Amazon S3 con wire tracing abilitato al momento della creazione del client.

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

In base al codice seguente e all'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 esiste, l'output è simile al seguente. (I ritorni sono stati aggiunti alla HEAD riga per motivi di 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/  
iAX5E30XRBzJ37cfc8T4D7ELC1KFELM=\r\n"  
-> "x-amz-request-id: 5MD4APQOS815QVBR\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 wire tracing dopo la creazione del client.

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

Per ulteriori informazioni sui campi delle informazioni relative al tracciamento bancario riportate, consulta [Transfer Family required request header](#).

## Aggiungere test con stubbing all'applicazione AWS SDK for Ruby

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'AWS SDK for Ruby disabilita il traffico di rete e il client restituisce dati inutilizzati (o falsi). Se non fornisci dati stubbati, il client restituisce:

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

L'esempio seguente restituisce nomi interi 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 tutti i dati inutilizzati, i valori predefiniti non sono più validi per gli attributi di istanza rimanenti. Ciò significa che nell'esempio precedente, l'attributo di istanza `rimanentecreation_date`, non è `manow. nil`

L' AWS SDK for Ruby convalida i dati bloccati. Se si trasmettono dati del tipo sbagliato, viene sollevata un'eccezione. `ArgumentError` Ad esempio, se invece dell'assegnazione precedente `abucket_data`, hai utilizzato quanto segue:

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

L' AWS SDK for Ruby `ArgumentError` solleva due eccezioni.

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

## Stubbing, errori del client

Puoi anche stub gli errori generati dall' AWS SDK for Ruby 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
```

## Utilizzo dei risultati impaginati nell' AWS SDK for Ruby

Molte AWS operazioni restituiscono risultati troncati quando il payload è troppo grande per essere restituito in un'unica risposta. Il servizio restituisce invece una parte dei dati e un token per recuperare il set successivo di elementi. Questo modello è noto come impaginazione.

### Le risposte paginate sono enumerabili

Il modo più semplice per gestire i dati di risposta paginati consiste nell'utilizzare l'enumeratore incorporato 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 ogni 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 impaginate

Per gestire autonomamente l'impaginazione, usa il `next_page?` metodo della risposta per verificare che ci siano altre pagine da recuperare oppure usa il `last_page?` metodo per verificare che non ci siano altre pagine da recuperare.

Se sono presenti più pagine, utilizzate il metodo `next_page` (notice there is no?) per recuperare la pagina successiva di risultati, come illustrato 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' AWS SDK for Ruby sono gestiti dalla classe [Aws::SeahorsePageableResponse::Client::Response](#) per fornire l'accesso ai dati paginati.

## Usare i camerieri nell' AWS SDK for Ruby

I camerieri sono metodi di utilità che verificano il verificarsi di uno stato particolare su un cliente. I camerieri possono fallire dopo diversi tentativi a un intervallo di polling definito per il cliente 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

## Invocare un cameriere

Per invocare un cameriere, chiama `wait_until` un cliente del servizio. Nell'esempio seguente, un cameriere attende che l'istanza `i-12345678` sia in esecuzione 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 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 richiamati per ciascuna operazione, consultate la documentazione `wait_until` sul campo `waiter_names` e relativa al client che state utilizzando.

## Attendi gli errori

I camerieri possono fallire con una delle seguenti eccezioni.

### [As: :Waiters: :Errors: FailureStateError](#)

È stato rilevato uno stato di errore durante l'attesa.

### [As: :Waiters: :Errors: NoSuchWaiterError](#)

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

### [As: :Waiters: :Errors: TooManyAttemptsError](#)

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

### [As: :Waiters: :Errors: UnexpectedError](#)

Si è verificato un errore imprevisto durante l'attesa.

### [As: :Waiters: :Errors: WaiterFailed](#)

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

Tutti questi errori, tranne, si basano `NoSuchWaiterError` su `WaiterFailed`. Per catturare gli errori in un cameriere, usa `WaiterFailed`, come mostrato 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 riprendere il controllo del programma. Per impostare questi valori, utilizzate i `delay` parametri `max_attempts` and nella chiamata `wait_until`. L'esempio seguente attende fino a 25 secondi, con un polling 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`.

## Allungare un cameriere

Per modificare il comportamento dei camerieri, è possibile registrare le callback che vengono attivate prima di ogni tentativo di sondaggio e prima di attendere.

L'esempio seguente implementa un backoff esponenziale in un cameriere raddoppiando il tempo di attesa per 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
```

# Esempi di codice SDK for Ruby

Gli esempi di codice riportati in questo argomento mostrano come utilizzare AWS SDK per Ruby with AWS.

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Alcuni servizi contengono ulteriori categorie di esempi che mostrano come sfruttare le librerie o le funzioni specifiche del servizio.

## Servizi

- [Esempi per Aurora con SDK per Ruby](#)
- [Esempi per Auto Scaling con SDK per Ruby](#)
- [CloudTrail esempi che utilizzano SDK for Ruby](#)
- [CloudWatch esempi che utilizzano SDK for Ruby](#)
- [Esempi per il gestore dell'identità per Amazon Cognito con SDK per Ruby](#)
- [Esempi per Amazon Comprehend con SDK per Ruby](#)
- [Esempi per Amazon DocumentDB con SDK per Ruby](#)
- [Esempi per DynamoDB con SDK per Ruby](#)
- [EC2 Esempi di Amazon che utilizzano SDK for Ruby](#)
- [Esempi per Elastic Beanstalk con SDK per Ruby](#)
- [EventBridge esempi che utilizzano SDK for Ruby](#)
- [AWS Glue esempi che utilizzano SDK for Ruby](#)
- [Esempi per IAM con SDK per Ruby](#)
- [Esempi per Kinesis con SDK per Ruby](#)
- [AWS KMS esempi che utilizzano SDK for Ruby](#)
- [Esempi per Lambda con SDK per Ruby](#)

- [Esempi per Amazon MSK con SDK per Ruby](#)
- [Esempi per Amazon Polly con SDK per Ruby](#)
- [Esempi per Amazon RDS con SDK per Ruby](#)
- [Esempi per Amazon S3 con SDK per Ruby](#)
- [Esempi per Amazon SES con SDK per Ruby](#)
- [Esempi per l'API Amazon SES v2 con SDK per Ruby](#)
- [Esempi per Amazon SNS con SDK per Ruby](#)
- [Esempi per Amazon SQS con SDK per Ruby](#)
- [AWS STS esempi che utilizzano SDK for Ruby](#)
- [Esempi per Amazon Textract con SDK per Ruby](#)
- [Esempi per Amazon Translate con SDK per Ruby](#)

## Esempi per Aurora con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Aurora.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Nozioni di base](#)

### Nozioni di base

Hello Aurora

L'esempio di codice seguente mostra come iniziare a utilizzare Aurora.

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-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Per i dettagli sull'API, consulta [Descrivi DBClusters](#) in AWS SDK per Ruby API Reference.

## Esempi per Auto Scaling con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l' AWS SDK per Ruby Auto Scaling.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti


- [Nozioni di base](#)

## Nozioni di base

### Hello Auto Scaling

Il seguente esempio di codice mostra come iniziare a usare Auto Scaling.

## 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-autoscaling'
require 'logger'

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

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:           #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/#{group.max_size}/
#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end
```

```
if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Per i dettagli sull'API, consulta la [DescribeAutoScalingGroups](#) sezione AWS SDK per Ruby API Reference.

## CloudTrail esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby with CloudTrail.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

## Azioni

### CreateTrail

Il seguente esempio di codice mostra come utilizzare `CreateTrail`.

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 [CreateTrail](#) sezione AWS SDK per Ruby API Reference.

## DeleteTrail

Il seguente esempio di codice mostra come utilizzare `DeleteTrail`.

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 => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
```

```
end
```

- Per i dettagli sull'API, consulta la [DeleteTrail](#) sezione AWS SDK per Ruby API Reference.

## ListTrails

Il seguente esempio di codice mostra come utilizzare `ListTrails`.

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
end
```

- Per i dettagli sull'API, consulta la [ListTrails](#) sezione AWS SDK per Ruby API Reference.

## LookupEvents

Il seguente esempio di codice mostra come utilizzare `LookupEvents`.

## 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 [LookupEvents](#) sezione AWS SDK per 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 per Ruby with CloudWatch.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### DescribeAlarms

Il seguente esempio di codice mostra come utilizzare `DescribeAlarms`.

#### 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-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Per i dettagli sull'API, consulta la [DescribeAlarms](#) sezione AWS SDK per Ruby API Reference.

## DescribeAlarmsForMetric

Il seguente esempio di codice mostra come utilizzare `DescribeAlarmsForMetric`.

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](#).

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

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

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
      end
    end
  end
end
```

```
    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 [DescribeAlarmsForMetric](#) sezione AWS SDK per Ruby API Reference.

## DisableAlarmActions

Il seguente esempio di codice mostra come utilizzare `DisableAlarmActions`.

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

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'
```

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

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

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


run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la [DisableAlarmActions](#) sezione AWS SDK per Ruby API Reference.

## ListMetrics

Il seguente esempio di codice mostra come utilizzare `ListMetrics`.

## 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 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 [ListMetrics](#) sezione AWS SDK per Ruby API Reference.

## PutMetricAlarm

Il seguente esempio di codice mostra come utilizzare `PutMetricAlarm`.

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

```
#   'NumberOfObjects',
#   ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#   'AWS/S3',
#   'Average',
#   [
#     {
#       name: 'BucketName',
#       value: 'amzn-s3-demo-bucket'
#     },
#     {
#       name: 'StorageType',
#       value: 'AllStorageTypes'
#     }
#   ],
#   86_400,
#   'Count',
#   1,
#   1,
#   'GreaterThanThreshold'
# )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
```

```

    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end

```

- Per i dettagli sull'API, consulta la [PutMetricAlarm](#) sezione AWS SDK per Ruby API Reference.

## PutMetricData

Il seguente esempio di codice mostra come utilizzare `PutMetricData`.

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-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.

```

```
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- Per i dettagli sull'API, consulta la [PutMetricData](#) sezione AWS SDK per Ruby API Reference.

## Esempi per il gestore dell'identità per Amazon Cognito con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Cognito Identity Provider. AWS SDK per Ruby

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Nozioni di base](#)

### Nozioni di base

Ciao Amazon Cognito

L'esempio di codice seguente mostra come iniziare a utilizzare Amazon Cognito.

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-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
```

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

# Lists and prints all user pools associated with the AWS account.
def list_user_pools
  paginator = @client.list_user_pools(max_results: 10)
  user_pools = []
  paginator.each_page do |page|
    user_pools.concat(page.user_pools)
  end

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

end

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

- Per i dettagli sull'API, consulta la [ListUserPools](#) sezione AWS SDK per Ruby API Reference.

## Esempi per Amazon Comprehend con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Comprehend. AWS SDK per Ruby

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Scenari](#)

## Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

### 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

# Esempi per Amazon DocumentDB con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon DocumentDB. AWS SDK per Ruby

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Esempi serverless](#)

## Esempi serverless

Invocare una funzione Lambda da un trigger Amazon DocumentDB

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

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 Amazon DocumentDB con Lambda tramite Ruby.

```
require 'json'

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

def log_document_db_event(record)
```

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

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

## Esempi per DynamoDB con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con DynamoDB.

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Nozioni di base](#)
- [Nozioni di base](#)
- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Nozioni di base

### Hello DynamoDB

L'esempio di codice seguente mostra come iniziare a utilizzare DynamoDB.

### 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-dynamodb'
require 'logger'

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

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

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

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

    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
    end
  end
end
```

```
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

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

- Per i dettagli sull'API, consulta la [ListTables](#) sezione AWS SDK per Ruby API Reference.

## Nozioni di base

### Informazioni di base

L'esempio di codice seguente mostra come:

- Crea una tabella in grado di contenere i dati del filmato.
- Inserire, ottenere e aggiornare un singolo filmato nella tabella.
- Scrivere i dati del filmato nella tabella da un file JSON di esempio.
- Eseguire 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' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end

```

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 unless continue.eql?('y')
  end
end
print "\nDone!\n".green

new_step(6, 'Scan for a batch of items using a filter expression.')
```

```
years = {}
years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
2018:')
years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
releases = dynamodb_wrapper.scan_items(years)
if !releases.empty?
  puts("Found #{releases.length} movies.")
  count = Question.ask(
    'How many do you want to see? ', method(:is_int), in_range(1, releases.length)
  )
  puts("Here are your #{count} movies:")
  releases.take(count).each do |release|
    puts("\t#{release['title']}")
  end
else
  puts("I don't know about any movies released between #{years[:start]} "\
    "and #{years[:end]}.")
end
print "\nDone!\n".green

new_step(7, 'Delete an item from the DynamoDB table.')
answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
")
if answer.eql?('y')
  dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
  puts("Removed '#{my_movie[:title]}' from the table.")
  print "\nDone!\n".green
end

new_step(8, 'Delete the DynamoDB table.')
answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
if answer.eql?('y')
  scaffold.delete_table
  puts("Deleted #{table_name}.")
else
  puts("Don't forget to delete the table when you're done!")
end
print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
  true
end
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Ruby .
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Azioni

### BatchExecuteStatement

Il seguente esempio di codice mostra come usare `BatchExecuteStatement`.

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](#).

Lettura di un batch di elementi mediante PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

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

```

    @table = @dynamodb.table(table_name)
  end

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

```

### Eliminazione di un batch di elementi mediante PartiQL.

```

class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

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

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

```

- Per i dettagli sull'API, consulta la [BatchExecuteStatements](#) sezione AWS SDK per Ruby API Reference.

## BatchWriteItem

Il seguente esempio di codice mostra come utilizzare `BatchWriteItem`.

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, :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 } })
    end
  end
end
```

```

    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 [BatchWriteItem](#) sezione AWS SDK per Ruby API Reference.

## CreateTable

Il seguente esempio di codice mostra come utilizzare CreateTable.

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, :table_name, :table

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

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #

```

```

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

```

- Per i dettagli sull'API, consulta la [CreateTable](#) sezione AWS SDK per Ruby API Reference.

## DeleteItem

Il seguente esempio di codice mostra come utilizzare `DeleteItem`.

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, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')

```

```

    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

```

- Per i dettagli sull'API, consulta la [DeleteItem](#) sezione AWS SDK per Ruby API Reference.

## DeleteTable

Il seguente esempio di codice mostra come utilizzare DeleteTable.

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, :table_name, :table

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

```

```
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 [DeleteTable](#) sezione AWS SDK per Ruby API Reference.

## DescribeTable

Il seguente esempio di codice mostra come utilizzare `DescribeTable`.

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, :table_name, :table

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

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

```

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

```

- Per i dettagli sull'API, consulta la [DescribeTable](#) sezione AWS SDK per Ruby API Reference.

## ExecuteStatement

Il seguente esempio di codice mostra come utilizzare `ExecuteStatement`.

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](#).

Selezione di un elemento mediante PartiQL.

```

class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

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

```

### Aggiunta di un elemento mediante PartiQL.

```

class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

```

## Eliminazione di un elemento mediante PartiQL.

```

class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)

```

```

request = {
  statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
  parameters: [title, year]
}
@dynamodb.client.execute_statement(request)
end

```

- Per i dettagli sull'API, consulta la [ExecuteStatement](#) sezione AWS SDK per Ruby API Reference.

## GetItem

Il seguente esempio di codice mostra come utilizzare `GetItem`.

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, :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")
  end
end

```

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

- Per i dettagli sull'API, consulta la [GetItem](#) sezione AWS SDK per Ruby API Reference.

## ListTables

Il seguente esempio di codice mostra come utilizzare `ListTables`.

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](#).

Determina se esiste una tabella.

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

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

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  end
end
```

```

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 [ListTables](#) sezione AWS SDK per Ruby API Reference.

## PutItem

Il seguente esempio di codice mostra come utilizzare `PutItem`.

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, :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] }
      }
    )
  end
end

```

```

    }
  )
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 [PutItem](#) sezione AWS SDK per Ruby API Reference.

## Query

Il seguente esempio di codice mostra come utilizzare `Query`.

### 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, :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 }
    )
  end
end

```

```
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 informazioni dettagliate sull'API, consulta [Query](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## Scan

Il seguente esempio di codice mostra come usare Scan.

### 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, :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 sull'API, consulta [Scan](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## UpdateItem

Il seguente esempio di codice mostra come usare `UpdateItem`.

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, :table

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

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

```

- Per i dettagli sull'API, consulta la [UpdateItem](#) sezione AWS SDK per Ruby API Reference.


## Scenari

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 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 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.')
```

```
return unless scaffold.exists?(table_name)

scaffold.delete_table
end
```

- Per i dettagli sull'API, consulta la [BatchExecuteStatement](#) sezione AWS SDK per 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.to_s.yellow
print "\n\nDone!\n".green

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

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

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

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

scaffold.delete_table
end
```

- Per i dettagli sull'API, consulta la [ExecuteStatement](#) sezione AWS SDK per Ruby API Reference.

## Esempi serverless

### Invocare una funzione Lambda da un trigger DynamoDB

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DynamoDB 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](#).

### Utilizzo di un evento DynamoDB con Lambda tramite Ruby.

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

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


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

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

### Segnalazione di errori di elementi batch per funzioni Lambda con un trigger DynamoDB

L'esempio di codice seguente mostra come implementare una risposta batch parziale per funzioni Lambda che ricevono eventi da un flusso DynamoDB. 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 dell'altro 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 DynamoDB con Lambda tramite Ruby.

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

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

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

## EC2 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 per Ruby con Amazon EC2.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Nozioni di base](#)
- [Azioni](#)

## Nozioni di base

Ciao Amazon EC2

Il seguente esempio di codice mostra come iniziare a usare 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'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end
end
```

```
end

private

# Fetches all EC2 instances using pagination.
#
# @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
def fetch_instances
  paginator = @client.describe_instances
  instances = []

  paginator.each_page do |page|
    page.reservations.each do |reservation|
      reservation.instances.each do |instance|
        instances << instance
      end
    end
  end

  instances
end

# Prints details of the given EC2 instances.
#
# @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
print.
def print_instances(instances)
  instances.each do |instance|
    @logger.info("Instance ID: #{instance.instance_id}")
    @logger.info("Instance Type: #{instance.instance_type}")
    @logger.info("Public IP: #{instance.public_ip_address}")
    @logger.info("Public DNS Name: #{instance.public_dns_name}")
    @logger.info("\n")
  end
end

end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- Per i dettagli sull'API, [DescribeSecurityGroups](#) consulta AWS SDK per Ruby API Reference.

## Azioni

### AllocateAddress

Il seguente esempio di codice mostra come utilizzare `AllocateAddress`.

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 Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- Per i dettagli sull'API, [AllocateAddress](#) consulta AWS SDK per Ruby API Reference.

### AssociateAddress

Il seguente esempio di codice mostra come utilizzare `AssociateAddress`.

## 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](#).

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The association ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- Per i dettagli sull'API, [AssociateAddress](#) consulta AWS SDK per Ruby API Reference.

## CreateKeyPair

Il seguente esempio di codice mostra come utilizzare `CreateKeyPair`.

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}'."
  true
end
```

```

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

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

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)

```

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

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, [CreateKeyPair](#) consulta AWS SDK per Ruby API Reference.

## CreateRouteTable

Il seguente esempio di codice mostra come utilizzare `CreateRouteTable`.

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:
#
# - 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
      }
    ]
  )
end
```

```

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

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

```

```
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, [CreateRouteTable](#) consulta AWS SDK per Ruby API Reference.

## CreateSecurityGroup

Il seguente esempio di codice mostra come utilizzare `CreateSecurityGroup`.

## 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
  )
end
```

```
puts "Created security group '#{group_name}' with ID " \
     "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
end
```

```

    }
  ]
}
]
)
puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

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

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

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

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

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

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

```

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

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

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

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

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

    return if sg.ip_permissions_egress.empty?

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

```
    end
  end

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

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

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

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

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

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

```
    ARGV
  end
end

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

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

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

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

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

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

def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp', '80',
    '80',
```

```
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [CreateSecurityGroup](#) consulta AWS SDK per Ruby API Reference.

## CreateSubnet

Il seguente esempio di codice mostra come utilizzare `CreateSubnet`.

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'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
```

```
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
```

```
availability_zone = ''
tag_key = ''
tag_value = ''
region = ''
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage:  ruby ec2-ruby-example-create-subnet.rb ' \
        'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
        'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-6713dfEX'
  cidr_block = '10.0.0.0/24'
  availability_zone = 'us-west-2a'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  cidr_block = ARGV[1]
  availability_zone = ARGV[2]
  tag_key = ARGV[3]
  tag_value = ARGV[4]
  region = ARGV[5]
end

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

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

```
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [CreateSubnet](#) consulta AWS SDK per Ruby API Reference.

## CreateVpc

Il seguente esempio di codice mostra come utilizzare `CreateVpc`.

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'

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
```

```
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

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

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

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

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''

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

```

```
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [CreateVpc](#) consulta AWS SDK per Ruby API Reference.

## DescribeInstances

Il seguente esempio di codice mostra come utilizzare `DescribeInstances`.

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'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
```

```
# @example
# list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [DescribeInstances](#) consulta AWS SDK per Ruby API Reference.

## DescribeRegions

Il seguente esempio di codice mostra come utilizzare `DescribeRegions`.

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'

# @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, [DescribeRegions](#) consulta AWS SDK per Ruby API Reference.

## ReleaseAddress

Il seguente esempio di codice mostra come utilizzare `ReleaseAddress`.

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](#).

```

# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end

```

- Per i dettagli sull'API, [ReleaseAddress](#) consulta AWS SDK per Ruby API Reference.

## StartInstances

Il seguente esempio di codice mostra come utilizzare `StartInstances`.

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'

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

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

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

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

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

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

  puts 'Could not start instance.'
end


run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [StartInstances](#) consulta AWS SDK per Ruby API Reference.

## StopInstances

Il seguente esempio di codice mostra come utilizzare `StopInstances`.

## 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 stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

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

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

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

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

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

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [StopInstances](#) consulta AWS SDK per Ruby API Reference.

## TerminateInstances

Il seguente esempio di codice mostra come utilizzare `TerminateInstances`.

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.'
```

```

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

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

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

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

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, [TerminateInstances](#) consulta AWS SDK per Ruby API Reference.

# Esempi per Elastic Beanstalk con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Elastic Beanstalk.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

## Azioni

### **DescribeApplications**

Il seguente esempio di codice mostra come utilizzare `DescribeApplications`

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)
    end
  end
end
```

```
    list_environments(application.application_name)
  end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:          #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:          #{env.cname}")
      @logger.info("    Health:       #{env.health}")
    end
    rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
      @logger.error("Error listing environments for application #{application_name}:
#{e.message}")
    end
  end
end
```

- Per i dettagli sull'API, [DescribeApplications](#) consulta AWS SDK per Ruby API Reference.

## ListAvailableSolutionStacks

Il seguente esempio di codice mostra come utilizzare `ListAvailableSolutionStacks`.

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 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, [ListAvailableSolutionStacks](#) consulta AWS SDK per Ruby API Reference.

## UpdateApplication

Il seguente esempio di codice mostra come utilizzare `UpdateApplication`.

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, [UpdateApplication](#) consulta AWS SDK per 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 per Ruby with EventBridge.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Scenari](#)

## 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
  false
end
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?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
end

```

```
    false
  rescue StandardError => e
    puts "Topic not found: #{e.message}"
    false
  end
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.'"
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  '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
  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?
    response = response.next_page
    next unless response.roles.count.positive?

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

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

```
end
```

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

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

```

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

```

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.'
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  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.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  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
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

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

```

Riavvia un'istanza Amazon Elastic Compute Cloud (Amazon EC2) e aggiunge informazioni sull'attività correlata a un flusso di log in CloudWatch Logs.

```

# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'

```

```
# )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''

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

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

  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
end
```

```

log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  "Error stopping or starting instance '#{instance_id}': #{e.message}",
  sequence_token
)
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
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

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per 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 per Ruby with AWS Glue.

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Nozioni di base](#)
- [Nozioni di base](#)
- [Azioni](#)

## Nozioni di base

### Salve AWS Glue

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Glue.

### 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-glue'
require 'logger'

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

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
```

```
@logger.info('Here are the Glue jobs in your account:')

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

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

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

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

- Per i dettagli sull'API, [ListJobs](#) consulta AWS SDK per Ruby API Reference.

## Nozioni di base

### Informazioni di base

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}")
end
```

```
    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'
      }
    )
  end
end
```

```
    },
    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
  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)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end
end

```

```
private

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

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

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

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

def run_job(wrapper, job_name, db_name)
  new_step(5, 'Run the job.')
```

```

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

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

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

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

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

  puts 'Demo complete.'
end

```

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 dell'API AWS SDK per Ruby .
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)

- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## Azioni

### CreateCrawler

Il seguente esempio di codice mostra come utilizzare `CreateCrawler`.

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, [CreateCrawler](#) consulta AWS SDK per Ruby API Reference.

## CreateJob

Il seguente esempio di codice mostra come utilizzare `CreateJob`.

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
end
```

- Per i dettagli sull'API, [CreateJob](#) consulta AWS SDK per Ruby API Reference.

## DeleteCrawler

Il seguente esempio di codice mostra come utilizzare `DeleteCrawler`.

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

- Per i dettagli sull'API, [DeleteCrawler](#) consulta AWS SDK per Ruby API Reference.

## DeleteDatabase

Il seguente esempio di codice mostra come utilizzare `DeleteDatabase`.

## 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


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

- Per i dettagli sull'API, [DeleteDatabase](#) consulta AWS SDK per Ruby API Reference.

## DeleteJob

Il seguente esempio di codice mostra come utilizzare `DeleteJob`.

## 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
end
```

- Per i dettagli sull'API, [DeleteJob](#) consulta AWS SDK per Ruby API Reference.

## DeleteTable

Il seguente esempio di codice mostra come utilizzare `DeleteTable`.

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

- Per i dettagli sull'API, [DeleteTable](#) consulta AWS SDK per Ruby API Reference.

## GetCrawler

Il seguente esempio di codice mostra come utilizzare `GetCrawler`.

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

- Per i dettagli sull'API, [GetCrawler](#) consulta AWS SDK per Ruby API Reference.

## GetDatabase

Il seguente esempio di codice mostra come utilizzare `GetDatabase`.

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

- Per i dettagli sull'API, [GetDatabase](#) consulta AWS SDK per Ruby API Reference.

## GetJobRun

Il seguente esempio di codice mostra come utilizzare `GetJobRun`.

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
end
```

- Per i dettagli sull'API, [GetJobRun](#) consulta AWS SDK per Ruby API Reference.

## GetJobRuns

Il seguente esempio di codice mostra come utilizzare `GetJobRuns`.

## 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
end
```

- Per i dettagli sull'API, [GetJobRuns](#) consulta AWS SDK per Ruby API Reference.

## GetTables

Il seguente esempio di codice mostra come utilizzare `GetTables`.

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

- Per i dettagli sull'API, [GetTables](#) consulta AWS SDK per Ruby API Reference.

## ListJobs

Il seguente esempio di codice mostra come utilizzare `ListJobs`.

## 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
end
```

- Per i dettagli sull'API, [ListJobs](#) consulta AWS SDK per Ruby API Reference.

## StartCrawler

Il seguente esempio di codice mostra come utilizzare `StartCrawler`.

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

- Per i dettagli sull'API, [StartCrawler](#) consulta AWS SDK per Ruby API Reference.

## StartJobRun

Il seguente esempio di codice mostra come utilizzare `StartJobRun`.

## 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
end
```

- Per i dettagli sull'API, [StartJobRun](#) consulta AWS SDK per Ruby API Reference.

## Esempi per IAM con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby with IAM.

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Nozioni di base](#)
- [Nozioni di base](#)
- [Azioni](#)

## Nozioni di base

Hello IAM

Il seguente esempio di codice mostra come iniziare a utilizzare 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](#).

```
require 'aws-sdk-iam'
require 'logger'

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

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

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

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- Per i dettagli sull'API, [ListPolicies](#) consulta AWS SDK per Ruby API Reference.

## Nozioni di base

### Informazioni di base

L'esempio di codice seguente 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 su 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
end
```

```
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!')
end

```

```
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 dell'API AWS SDK per Ruby .
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Azioni

### **AttachRolePolicy**

Il seguente esempio di codice mostra come usare `AttachRolePolicy`.

## 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, collega e scollega le policy 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
  end
end
```

```
@logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

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

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

# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
```

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

- Per i dettagli sull'API, [AttachRolePolicy](#) consulta AWS SDK per Ruby API Reference.

## AttachUserPolicy

Il seguente esempio di codice mostra come utilizzare `AttachUserPolicy`.

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](#).

```
# 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
```

```
end
```

- Per i dettagli sull'API, [AttachUserPolicy](#) consulta AWS SDK per Ruby API Reference.

## CreateAccessKey

Il seguente esempio di codice mostra come utilizzare `CreateAccessKey`.

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

```
    []
  end

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

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

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
```

```
@iam_client.delete_access_key(  
  user_name: user_name,  
  access_key_id: access_key_id  
)  
true  
rescue StandardError => e  
  @logger.error("Error deleting access key: #{e.message}")  
  false  
end  
end
```

- Per i dettagli sull'API, [CreateAccessKey](#) consulta AWS SDK per Ruby API Reference.

## CreateAccountAlias

Il seguente esempio di codice mostra come utilizzare `CreateAccountAlias`.

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](#).

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, [CreateAccountAlias](#) consulta AWS SDK per Ruby API Reference.

## CreatePolicy

Il seguente esempio di codice mostra come utilizzare `CreatePolicy`.

## 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, collega e scollega le policy 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
  end
end
```

```
@logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

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

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

# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
```

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

- Per i dettagli sull'API, [CreatePolicy](#) consulta AWS SDK per Ruby API Reference.

## CreateRole

Il seguente esempio di codice mostra come utilizzare `CreateRole`.

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
end

```

- Per i dettagli sull'API, [CreateRole](#) consulta AWS SDK per Ruby API Reference.

## CreateServiceLinkedRole

Il seguente esempio di codice mostra come utilizzare `CreateServiceLinkedRole`.

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}.")
end

```

```

    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
end

```

- Per i dettagli sull'API, [CreateServiceLinkedRole](#) consulta AWS SDK per Ruby API Reference.

## CreateUser

Il seguente esempio di codice mostra come utilizzare `CreateUser`.

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, [CreateUser](#) consulta AWS SDK per Ruby API Reference.

## DeleteAccessKey

Il seguente esempio di codice mostra come utilizzare `DeleteAccessKey`.

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, 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
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
  end
  []
end
```

```
rescue StandardError => e
  @logger.error("Error listing access keys: #{e.message}")
  []
end

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

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

# Deletes an access key
#
# @param user_name [String] The name of the user.
# @param access_key_id [String] The ID for the access key.
```

```
# @return [Boolean]
def delete_access_key(user_name, access_key_id)
  @iam_client.delete_access_key(
    user_name: user_name,
    access_key_id: access_key_id
  )
  true
rescue StandardError => e
  @logger.error("Error deleting access key: #{e.message}")
  false
end
end
```

- Per i dettagli sull'API, [DeleteAccessKey](#) consulta AWS SDK per Ruby API Reference.

## DeleteAccountAlias

Il seguente esempio di codice mostra come utilizzare `DeleteAccountAlias`.

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](#).

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
end


```

- Per i dettagli sull'API, [DeleteAccountAlias](#) consulta AWS SDK per Ruby API Reference.

## DeleteRole

Il seguente esempio di codice mostra come utilizzare `DeleteRole`.

## 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)
  # 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, [DeleteRole](#) consulta AWS SDK per Ruby API Reference.

## DeleteServerCertificate

Il seguente esempio di codice mostra come utilizzare `DeleteServerCertificate`.

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](#).

Elenca, aggiorna ed elimina i certificati 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
end


```

- Per i dettagli sull'API, [DeleteServerCertificate](#) consulta AWS SDK per Ruby API Reference.

## DeleteServiceLinkedRole

Il seguente esempio di codice mostra come utilizzare `DeleteServiceLinkedRole`.

## 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 service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id
    )
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)

    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
```

```
def handle_deletion_error(e, role_name)
  return if e.code == 'NoSuchEntity'

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

- Per i dettagli sull'API, [DeleteServiceLinkedRole](#) consulta AWS SDK per Ruby API Reference.

## DeleteUser

Il seguente esempio di codice mostra come utilizzare `DeleteUser`.

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, [DeleteUser](#) consulta AWS SDK per Ruby API Reference.

## DeleteUserPolicy

Il seguente esempio di codice mostra come utilizzare `DeleteUserPolicy`.

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, [DeleteUserPolicy](#) consulta AWS SDK per Ruby API Reference.

## DetachRolePolicy

Il seguente esempio di codice mostra come utilizzare `DetachRolePolicy`.

## 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, collega e scollega le policy 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
  end
end
```

```
@logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

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

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

# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
```

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

- Per i dettagli sull'API, [DetachRolePolicy](#) consulta AWS SDK per Ruby API Reference.

## DetachUserPolicy

Il seguente esempio di codice mostra come utilizzare `DetachUserPolicy`.

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, [DetachUserPolicy](#) consulta AWS SDK per Ruby API Reference.

## GetAccountPasswordPolicy

Il seguente esempio di codice mostra come utilizzare `GetAccountPasswordPolicy`.

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 IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

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

  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is: #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
    #{e.message}")
  end
end
```

```
    raise
  end
end
```

- Per i dettagli sull'API, [GetAccountPasswordPolicy](#) consulta AWS SDK per Ruby API Reference.

## GetPolicy

Il seguente esempio di codice mostra come utilizzare `GetPolicy`.

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, [GetPolicy](#) consulta AWS SDK per Ruby API Reference.

## GetRole

Il seguente esempio di codice mostra come utilizzare `GetRole`.

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, [GetRole](#) consulta AWS SDK per Ruby API Reference.

## GetUser

Il seguente esempio di codice mostra come utilizzare `GetUser`.

## 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](#).


```
# 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, [GetUser](#) consulta AWS SDK per Ruby API Reference.

## ListAccessKeys

Il seguente esempio di codice mostra come utilizzare `ListAccessKeys`.

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

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

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


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

- Per i dettagli sull'API, [ListAccessKeys](#) consulta AWS SDK per Ruby API Reference.

## ListAccountAliases

Il seguente esempio di codice mostra come utilizzare `ListAccountAliases`.

## 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](#).

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}")
  end
end
```

```

    false
  end

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

```

- Per i dettagli sull'API, [ListAccountAliases](#) consulta AWS SDK per Ruby API Reference.

## ListAttachedRolePolicies

Il seguente esempio di codice mostra come utilizzare `ListAttachedRolePolicies`.

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, collega e scollega le policy 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
end

```

```
end

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

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

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

```

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

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

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


```

- Per i dettagli sull'API, [ListAttachedRolePolicies](#) consulta AWS SDK per Ruby API Reference.

## ListGroups

Il seguente esempio di codice mostra come utilizzare `ListGroups`.

## 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](#).

```
# 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, [ListGroups](#) consulta AWS SDK per Ruby API Reference.

## ListPolicies

Il seguente esempio di codice mostra come utilizzare `ListPolicies`.

## 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, collega e scollega le policy 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
  end
end
```

```
@logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end

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

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

# Detaches a policy from a role
#
# @param role_name [String] The name of the role
# @param policy_arn [String] The policy ARN
```

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

- Per i dettagli sull'API, [ListPolicies](#) consulta AWS SDK per Ruby API Reference.

## ListRolePolicies

Il seguente esempio di codice mostra come utilizzare `ListRolePolicies`.

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 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, [ListRolePolicies](#) consulta AWS SDK per Ruby API Reference.

## ListRoles

Il seguente esempio di codice mostra come utilizzare `ListRoles`.

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, [ListRoles](#) consulta AWS SDK per Ruby API Reference.

## ListSAMLProviders

Il seguente esempio di codice mostra come utilizzare `ListSAMLProviders`.

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 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 i dettagli sull'API, consulta [List SAMLProviders](#) in AWS SDK per Ruby API Reference.

## ListServerCertificates

Il seguente esempio di codice mostra come utilizzare `ListServerCertificates`.

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](#).

Elenca, aggiorna ed elimina i certificati 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
end


```

- Per i dettagli sull'API, [ListServerCertificates](#) consulta AWS SDK per Ruby API Reference.

## ListUsers

Il seguente esempio di codice mostra come utilizzare `ListUsers`.

## 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 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, [ListUsers](#) consulta AWS SDK per Ruby API Reference.

## PutUserPolicy

Il seguente esempio di codice mostra come utilizzare `PutUserPolicy`.

## 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 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, [PutUserPolicy](#) consulta AWS SDK per Ruby API Reference.

## UpdateServerCertificate

Il seguente esempio di codice mostra come utilizzare `UpdateServerCertificate`.

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](#).

Elenca, aggiorna ed elimina i certificati server.

```

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

```

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, [UpdateServerCertificate](#) consulta AWS SDK per Ruby API Reference.

## UpdateUser

Il seguente esempio di codice mostra come utilizzare `UpdateUser`.

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}")
end

```

```
false  
end
```

- Per i dettagli sull'API, [UpdateUser](#) consulta AWS SDK per Ruby API Reference.

## Esempi per Kinesis con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando il AWS SDK per Ruby con Kinesis.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Esempi serverless](#)

## Esempi serverless

Invocare una funzione Lambda da un trigger Kinesis

L'esempio di codice seguente 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 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 Kinesis con Lambda tramite Ruby.

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

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

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

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

L'esempio di codice seguente mostra come implementare una risposta batch parziale per 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 dell'altro 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 Kinesis con Lambda tramite Ruby.

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

```
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

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

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

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

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

## AWS KMS esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby with AWS KMS.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Azioni](#)

## Azioni

### CreateKey

Il seguente esempio di codice mostra come utilizzare `CreateKey`.

#### 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, [CreateKey](#) consulta AWS SDK per Ruby API Reference.

## Decrypt

Il seguente esempio di codice mostra come utilizzare `Decrypt`.

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'

# 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 informazioni dettagliate sull'API, consulta [Decrypt](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## Encrypt

Il seguente esempio di codice mostra come usare `Encrypt`.

## 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'

# 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 informazioni dettagliate sull'API, consulta [Encrypt](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## ReEncrypt

Il seguente esempio di codice mostra come usare `ReEncrypt`.

## 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'

# 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, [ReEncrypt](#) consulta AWS SDK per Ruby API Reference.

## Esempi per Lambda con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby with Lambda.

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Nozioni di base](#)
- [Nozioni di base](#)
- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Nozioni di base

Hello Lambda

Il seguente esempio di codice mostra come iniziare a usare 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](#).

```
require 'aws-sdk-lambda'  
  
# Creates an AWS Lambda client using the default credentials and configuration  
def lambda_client
```

```
Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
# necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end

  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Per i dettagli sull'API, [ListFunctions](#) consulta AWS SDK per Ruby API Reference.

## Nozioni di base

### Informazioni di base

L'esempio di codice seguente mostra come:

- Crea un ruolo IAM e una funzione Lambda, quindi carica il codice del gestore.
- Invocare la funzione con un singolo parametro e ottenere i risultati.
- Aggiorna il codice della funzione e configuralo con una variabile di ambiente.
- Invocare 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, 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)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
  role_policy = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Principal': { 'Service': 'lambda.amazonaws.com' },
        'Action': 'sts:AssumeRole'
      }
    ]
  }
end
```

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

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

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

Definisci un gestore Lambda che incrementa un numero fornito come parametro di invocazione.

```
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.
#
# @param source_file: The name of the object, without suffix, for the Lambda file
and zip.
# @return: The deployment package.
def create_deployment_package(source_file)
  Dir.chdir(File.dirname(__FILE__))
  if File.exist?('lambda_function.zip')
    File.delete('lambda_function.zip')
    @logger.debug('Deleting old zip: lambda_function.zip')
  end
  Zip::File.open('lambda_function.zip', create: true) do |zipfile|
    zipfile.add('lambda_function.rb', "#{source_file}.rb")
  end
end

```

```
@logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
File.read('lambda_function.zip').to_s
rescue StandardError => e
  @logger.error("There was an error creating deployment package:\n #{e.message}")
end
```

## 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.
# @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
```

Invocare 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
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
end
```

```
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\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 dell'API AWS SDK per Ruby .
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Azioni

### CreateFunction

Il seguente esempio di codice mostra come usare `CreateFunction`.

## 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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

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

```

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

```

- Per i dettagli sull'API, [CreateFunction](#) consulta AWS SDK per Ruby API Reference.

## DeleteFunction

Il seguente esempio di codice mostra come utilizzare `DeleteFunction`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)

```

```

print "Deleting function: #{function_name}..."
@lambda_client.delete_function(
  function_name: function_name
)
print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end

```

- Per i dettagli sull'API, [DeleteFunction](#) consulta AWS SDK per Ruby API Reference.

## GetFunction

Il seguente esempio di codice mostra come utilizzare `GetFunction`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {

```

```

        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
end

```

- Per i dettagli sull'API, [GetFunction](#) consulta AWS SDK per Ruby API Reference.

## Invoke

Il seguente esempio di codice mostra come utilizzare `Invoke`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  end
end

```

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

- Per informazioni dettagliate sull'API, consulta [Invoke](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## ListFunctions

Il seguente esempio di codice mostra come usare `ListFunctions`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  end
rescue Aws::Lambda::Errors::ServiceException => e
```

```
@logger.error("There was an error listing functions:\n #{e.message}")
end
```

- Per i dettagli sull'API, [ListFunctions](#) consulta AWS SDK per Ruby API Reference.

## UpdateFunctionCode

Il seguente esempio di codice mostra come utilizzare `UpdateFunctionCode`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Updates the code for a Lambda function by submitting a .zip archive that
  # contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                             .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
  end
end
```

```

    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    rescue Aws::Lambda::Errors::ServiceException => e
      @logger.error("There was an error updating 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, [UpdateFunctionCode](#) consulta AWS SDK per Ruby API Reference.

## UpdateFunctionConfiguration

Il seguente esempio di codice mostra come utilizzare `UpdateFunctionConfiguration`.

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 LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.

```

```
# @return: Data about the update, including the status.
def update_function_configuration(function_name, log_level)
  @lambda_client.update_function_configuration({
    function_name: function_name,
    environment: {
      variables: {
        'LOG_LEVEL' => log_level
      }
    }
  })

  @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
    w.max_attempts = 5
    w.delay = 5
  end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Writers::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
end
```

- Per i dettagli sull'API, [UpdateFunctionConfiguration](#) consulta AWS SDK per Ruby API Reference.

## Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

### 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

## Esempi serverless

Connessione a un database Amazon RDS in una funzione Lambda

L'esempio di codice seguente mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

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](#).

Connessione a un database Amazon RDS in una funzione Lambda tramite Ruby.

```
# Ruby code here.  
  
require 'aws-sdk-rds'  
require 'json'  
require 'mysql2'
```

```
def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
  {
    statusCode: 200,
    body: result.to_json
  }
  rescue => e
```

```
    puts "Database connection failed due to #{e}"
  end
end
```

## Invocare una funzione Lambda da un trigger Kinesis

L'esempio di codice seguente 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 dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Utilizzo di un evento Kinesis con Lambda tramite Ruby.

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

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

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
```

```
# Placeholder for actual async work
# You can use Ruby's asynchronous programming tools like async/await or fibers
here.
return data
end
```

## Invocare una funzione Lambda da un trigger DynamoDB

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso DynamoDB. La funzione recupera il payload DynamoDB e registra il contenuto del record.

### SDK per Ruby

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento DynamoDB con Lambda tramite Ruby.

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

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

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

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

## Invocare una funzione Lambda da un trigger Amazon DocumentDB

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un flusso di modifiche di DocumentDB. La funzione recupera il payload DocumentDB e registra il contenuto del record.

### SDK per Ruby

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento Amazon DocumentDB con Lambda tramite Ruby.

```
require 'json'

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

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

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

## Invocare una funzione Lambda da un trigger Amazon MSK

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK e registra il contenuto del record.

### SDK per Ruby

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento Amazon MSK con Lambda tramite Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"


    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

## Invocazione di una funzione Lambda da un trigger Amazon S3

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

## SDK per Ruby

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

## Utilizzo di un evento S3 con Lambda tramite Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
  Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

## Invocazione di una funzione Lambda da un trigger Amazon SNS

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dal ricevimento 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 dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento SNS con Lambda tramite Ruby.


```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

## Invocazione di una funzione Lambda da un trigger Amazon SQS

L'esempio di codice seguente 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 dell'altro 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.


```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Kinesis

L'esempio di codice seguente mostra come implementare una risposta batch parziale per 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 dell'altro 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 Kinesis con Lambda tramite Ruby.

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

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

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

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

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

```
end
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger DynamoDB

L'esempio di codice seguente mostra come implementare una risposta batch parziale per funzioni Lambda che ricevono eventi da un flusso DynamoDB. 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 dell'altro 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 DynamoDB con Lambda tramite Ruby.

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

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

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

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

L'esempio di codice seguente mostra come implementare una risposta batch parziale per 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 dell'altro 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.

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

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
        rescue StandardError => e
          batch_item_failures << {"itemIdentifier" => record['messageId']}
        end
      end

      sqs_batch_response["batchItemFailures"] = batch_item_failures
      return sqs_batch_response
    end
  end
end
```

# Esempi per Amazon MSK con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Amazon MSK.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Esempi serverless](#)

## Esempi serverless

Invocare una funzione Lambda da un trigger Amazon MSK

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di record da un cluster Amazon MSK. La funzione recupera il payload MSK e registra il contenuto del record.

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 Amazon MSK con Lambda tramite Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
    end
  end
end
```

```
# Decode base64
msg = Base64.decode64(record['value'])
puts "Message: #{msg}"
end
end
end
```

## Esempi per Amazon Polly con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Polly. AWS SDK per Ruby

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)
- [Scenari](#)

## Azioni

### **DescribeVoices**

Il seguente esempio di codice mostra come usare `DescribeVoices`.

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
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Per i dettagli sull'API, [DescribeVoices](#) consulta AWS SDK per Ruby API Reference.

## ListLexicons

Il seguente esempio di codice mostra come utilizzare `ListLexicons`.

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
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Per i dettagli sull'API, [ListLexicons](#) consulta AWS SDK per Ruby API Reference.

## SynthesizeSpeech

Il seguente esempio di codice mostra come utilizzare `SynthesizeSpeech`.

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 => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- Per i dettagli sull'API, [SynthesizeSpeech](#) consulta AWS SDK per Ruby API Reference.

## Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

### 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

## Esempi per Amazon RDS con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon RDS. AWS SDK per Ruby

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Nozioni di base](#)
- [Azioni](#)
- [Esempi serverless](#)

## Nozioni di base

Hello Amazon RDS

Il seguente esempio di codice mostra come iniziare a usare Amazon RDS.

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'
require 'logger'

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

  # Lists and prints all RDS DB instances in the current AWS account.
```

```
def list_db_instances
  @logger.info('Listing RDS DB instances')

  paginator = @client.describe_db_instances
  instances = []

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

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

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


- Per i dettagli sull'API, consulta [Descrivi DBInstances](#) in AWS SDK per Ruby API Reference.

## Azioni

### CreateDBSnapshot

Il seguente esempio di codice mostra come utilizzare `CreateDBSnapshot`.

## 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 i dettagli sull'API, consulta [Create DBSnapshot](#) in AWS SDK per Ruby API Reference.

## DescribeDBInstances

Il seguente esempio di codice mostra come utilizzare `DescribeDBInstances`.

## 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 i dettagli sull'API, consulta [Descrivi DBInstances](#) in AWS SDK per Ruby API Reference.

## DescribeDBParameterGroups

Il seguente esempio di codice mostra come utilizzare `DescribeDBParameterGroups`.

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) 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 [Descrivere DBParameter i gruppi](#) in AWS SDK per Ruby API Reference.

## DescribeDBParameters

Il seguente esempio di codice mostra come utilizzare `DescribeDBParameters`.

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) 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

```

```
end
parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Per i dettagli sull'API, consulta [Descrivi DBParameters](#) in AWS SDK per Ruby API Reference.

## DescribeDBSnapshots

Il seguente esempio di codice mostra come utilizzare `DescribeDBSnapshots`.

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 i dettagli sull'API, consulta [Descrivi DBSnapshots](#) in AWS SDK per Ruby API Reference.

## Esempi serverless

### Connessione a un database Amazon RDS in una funzione Lambda

L'esempio di codice seguente mostra come implementare una funzione Lambda che si connette a un database RDS. La funzione effettua una semplice richiesta al database e restituisce il risultato.

#### 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](#).

### Connessione a un database Amazon RDS in una funzione Lambda tramite Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']          # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']    # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
```

```
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_clear_text_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

## Esempi per Amazon S3 con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon S3. AWS SDK per Ruby

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Nozioni di base](#)
- [Nozioni di base](#)
- [Azioni](#)
- [Scenari](#)
- [Esempi serverless](#)

## Nozioni di base

Hello Amazon S3

Il seguente esempio di codice mostra come iniziare a usare Amazon 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](#).

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
  end
end
```

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

# Lists and prints all S3 buckets in the current AWS account.
def list_buckets
  @logger.info('Here are the buckets in your account:')

  response = @client.list_buckets

  if response.buckets.empty?
    @logger.info("You don't have any S3 buckets yet.")
  else
    response.buckets.each do |bucket|
      @logger.info("- #{bucket.name}")
    end
  end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per Ruby API Reference.

## Nozioni di base

### Informazioni di base

L'esempio di codice seguente mostra come:

- Creare un bucket e caricare 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 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 the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

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

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end
end
```

```
# 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
end

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

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

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

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

```
        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 dell'API AWS SDK per Ruby .
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)

- [PutObject](#)

## Azioni

### CopyObject

Il seguente esempio di codice mostra come usare `CopyObject`.

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)
  end
end
```

```

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

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

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

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 = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [CopyObject](#) consulta AWS SDK per Ruby API Reference.

## CreateBucket

Il seguente esempio di codice mostra come utilizzare `CreateBucket`.

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 BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  #
  #
  # create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    end
  end
end
```

```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, [CreateBucket](#) consulta AWS SDK per Ruby API Reference.

## DeleteBucket

Il seguente esempio di codice mostra come utilizzare `DeleteBucket`.

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, [DeleteBucket](#) consulta AWS SDK per Ruby API Reference.

## DeleteBucketCors

Il seguente esempio di codice mostra come utilizzare `DeleteBucketCors`.

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, [DeleteBucketCors](#) consulta AWS SDK per Ruby API Reference.

## DeleteBucketPolicy

Il seguente esempio di codice mostra come utilizzare `DeleteBucketPolicy`.

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
end

```

- Per i dettagli sull'API, [DeleteBucketPolicy](#) consulta AWS SDK per Ruby API Reference.

## DeleteObjects

Il seguente esempio di codice mostra come utilizzare `DeleteObjects`.

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, [DeleteObjects](#) consulta AWS SDK per Ruby API Reference.

## GetBucketCors

Il seguente esempio di codice mostra come utilizzare `GetBucketCors`.

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

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
  # for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
    nil
  end
end
```

- Per i dettagli sull'API, [GetBucketCors](#) consulta AWS SDK per Ruby API Reference.

## GetBucketPolicy

Il seguente esempio di codice mostra come utilizzare `GetBucketPolicy`.

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 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, [GetBucketPolicy](#) consulta AWS SDK per Ruby API Reference.

## GetObject

Il seguente esempio di codice mostra come utilizzare `GetObject`.

## 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](#).

## 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 = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

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

```

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

  run_demo if $PROGRAM_NAME == __FILE__

```

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

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

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

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
  obj_data.server_side_encryption

```

```
puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per Ruby API Reference.

## HeadObject

Il seguente esempio di codice mostra come utilizzare `HeadObject`.

### 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 object actions.
class ObjectExistsWrapper
  attr_reader :object

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

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
    Here's why: #{e.message}"
    false
  end
end
```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [HeadObject](#) consulta AWS SDK per Ruby API Reference.

## ListBuckets

Il seguente esempio di codice mostra come utilizzare `ListBuckets`.

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 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, [ListBuckets](#) consulta AWS SDK per Ruby API Reference.

## ListObjectsV2

Il seguente esempio di codice mostra come utilizzare `ListObjectsV2`.

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

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

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

run_demo if $PROGRAM_NAME == __FILE__


```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per Ruby API Reference.

## PutBucketCors

Il seguente esempio di codice mostra come utilizzare PutBucketCors.

## 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}"
  end
end
```

```

    false
  end

end

```

- Per i dettagli sull'API, [PutBucketCors](#) consulta AWS SDK per Ruby API Reference.

## PutBucketPolicy

Il seguente esempio di codice mostra come utilizzare `PutBucketPolicy`.

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 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, [PutBucketPolicy](#) consulta AWS SDK per Ruby API Reference.

## PutBucketWebsite

Il seguente esempio di codice mostra come utilizzare `PutBucketWebsite`.

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 website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
  # configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
  # errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
  # false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
  end
end
```

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

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK per Ruby API Reference.

## PutObject

Il seguente esempio di codice mostra come utilizzare `PutObject`.

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 = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

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

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```


- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per Ruby API Reference.

## Scenari

### Creazione di un URL prefirmato

L'esempio di codice seguente mostra come creare un URL prefirmato 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 = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

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

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

  case response
  when Net::HTTPSuccess
```

```
    puts 'Content uploaded!'
  else
    puts response.value
  end
end
end

run_demo if $PROGRAM_NAME == __FILE__
```

## Esempi serverless

### Invocazione di una funzione Lambda da un trigger Amazon S3

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

### SDK per Ruby

#### Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

### Utilizzo di un evento S3 con Lambda tramite Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
```

```
key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
Encoding::UTF_8)
begin
  response = s3.get_object(bucket: bucket, key: key)
  puts "CONTENT TYPE: #{response.content_type}"
  return response.content_type
rescue StandardError => e
  puts e.message
  puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
and your bucket is in the same region as this function."
  raise e
end
end
```

## Esempi per Amazon SES con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Amazon SES.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti


- [Azioni](#)

### Azioni

#### **GetIdentityVerificationAttributes**

Il seguente esempio di codice mostra come usare `GetIdentityVerificationAttributes`.

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: 'EmailAddress'
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status


  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Per i dettagli sull'API, [GetIdentityVerificationAttributes](#) consulta AWS SDK per Ruby API Reference.

## ListIdentities

Il seguente esempio di codice mostra come utilizzare `ListIdentities`.

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: 'EmailAddress'
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status


  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Per i dettagli sull'API, [ListIdentities](#) consulta AWS SDK per Ruby API Reference.

## SendEmail

Il seguente esempio di codice mostra come utilizzare `SendEmail`.

## 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-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>.'
```

```
# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Per i dettagli sull'API, [SendEmail](#) consulta AWS SDK per Ruby API Reference.

## VerifyEmailIdentity

Il seguente esempio di codice mostra come utilizzare `VerifyEmailIdentity`.

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

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Per i dettagli sull'API, [VerifyEmailIdentity](#) consulta AWS SDK per Ruby API Reference.

# Esempi per l'API Amazon SES v2 con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando l'API AWS SDK per Ruby with Amazon SES v2.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti

- [Azioni](#)

## Azioni

### SendEmail

Il seguente esempio di codice mostra come usare `SendEmail`.

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-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, [SendEmail](#) consulta AWS SDK per Ruby API Reference.

## Esempi per Amazon SNS con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Amazon SNS.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Azioni](#)
- [Esempi serverless](#)

# Azioni

## CreateTopic

Il seguente esempio di codice mostra come usare `CreateTopic`.

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 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
end
```

```
sns_topic_creator = SNS::TopicCreator.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 per Ruby](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per Ruby API Reference.

## ListSubscriptions

Il seguente esempio di codice mostra come utilizzare `ListSubscriptions`.

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 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
  end
end
```

```
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
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
end
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK per Ruby API Reference.

## ListTopics

Il seguente esempio di codice mostra come utilizzare `ListTopics`.

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

```
def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK per Ruby API Reference.

## Publish

Il seguente esempio di codice mostra come utilizzare `Publish`.

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](#).

```
# 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 per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## SetTopicAttributes

Il seguente esempio di codice mostra come usare `SetTopicAttributes`.

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](#).

```
# 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

# 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 per Ruby](#).
- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK per Ruby API Reference.

## Subscribe

Il seguente esempio di codice mostra come utilizzare `Subscribe`.

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](#).

Sottoscrive un indirizzo e-mail a un argomento.

```
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
```

```
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
end
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

## Esempi serverless

Invocazione di una funzione Lambda da un trigger Amazon SNS

L'esempio di codice seguente mostra come implementare una funzione Lambda che riceve un evento attivato dal ricevimento 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 dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

## Esempi per Amazon SQS con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby con Amazon SQS.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti


- [Azioni](#)
- [Esempi serverless](#)

## Azioni

### **ChangeMessageVisibility**

Il seguente esempio di codice mostra come usare `ChangeMessageVisibility`.

## 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 up to 10 messages
  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "Before attempting to change message visibility timeout: received
#{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not
be visible for 30 seconds after first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })
```

```
puts "\nAfter attempting to change message visibility timeout: received
#{receive_message_result_after.messages.count} message(s)."
```

```
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
  exist."
end
```

- Per i dettagli sull'API, [ChangeMessageVisibility](#) consulta AWS SDK per Ruby API Reference.

## CreateQueue

Il seguente esempio di codice mostra come utilizzare `CreateQueue`.

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, [CreateQueue](#) consulta AWS SDK per Ruby API Reference.

## DeleteQueue

Il seguente esempio di codice mostra come utilizzare `DeleteQueue`.

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, [DeleteQueue](#) consulta AWS SDK per Ruby API Reference.

## ListQueues

Il seguente esempio di codice mostra come utilizzare `ListQueues`.

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.
# @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, [ListQueues](#) consulta AWS SDK per Ruby API Reference.

## ReceiveMessage

Il seguente esempio di codice mostra come utilizzare `ReceiveMessage`.

## 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
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, [ReceiveMessage](#) consulta AWS SDK per Ruby API Reference.

## SendMessage

Il seguente esempio di codice mostra come utilizzare `SendMessage`.

## 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, [SendMessage](#) consulta AWS SDK per Ruby API Reference.

## SendMessageBatch

Il seguente esempio di codice mostra come utilizzare `SendMessageBatch`.

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 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, [SendMessageBatch](#) consulta AWS SDK per Ruby API Reference.

## Esempi serverless

### Invocazione di una funzione Lambda da un trigger Amazon SQS

L'esempio di codice seguente 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 altro 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.

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

```
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

## Segnalazione di errori di elementi batch per funzioni Lambda con un trigger Amazon SQS

L'esempio di codice seguente mostra come implementare una risposta batch parziale per 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 dell'altro 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.

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

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

```
sqs_batch_response = {}

event["Records"].each do |record|
  begin
    # process message
    rescue StandardError => e
      batch_item_failures << {"itemIdentifier" => record['messageId']}
    end
  end

  sqs_batch_response["batchItemFailures"] = batch_item_failures
  return sqs_batch_response
end
end
```

## AWS STS esempi che utilizzano SDK for Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando AWS SDK per Ruby with AWS STS.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Ogni esempio include un link al codice sorgente completo, in cui vengono fornite le istruzioni su come configurare ed eseguire il codice nel contesto.

Argomenti


- [Azioni](#)

### Azioni

#### **AssumeRole**

Il seguente esempio di codice mostra come utilizzare `AssumeRole`.

## 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, [AssumeRole](#) consulta AWS SDK per Ruby API Reference.

# Esempi per Amazon Textract con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Textract. AWS SDK per Ruby

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice nel contesto.

## Argomenti

- [Scenari](#)

## Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

### 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

## Esempi per Amazon Translate con SDK per Ruby

I seguenti esempi di codice mostrano come eseguire azioni e implementare scenari comuni utilizzando Amazon Translate. AWS SDK per Ruby

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Ogni esempio include un link al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice nel contesto.

### Argomenti

- [Scenari](#)

## Scenari

Crea un'applicazione per analizzare il feedback dei clienti

L'esempio di codice seguente mostra come creare un'applicazione che analizza le schede dei commenti dei clienti, le traduce dalla loro lingua originale, ne determina il sentiment e genera un file audio dal testo tradotto.

### 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

# Migrazione da AWS SDK for Ruby versione 1 o 2 a SDK AWS for Ruby versione 3

Questo argomento include dettagli per aiutarti a migrare dalla versione 1 o 2 dell' AWS SDK for Ruby alla versione 3.

## Side-by-side utilizzo

Non è necessario sostituire la versione 1 o 2 dell' AWS SDK for Ruby con la versione 3. È possibile utilizzarli insieme nella stessa applicazione. Per ulteriori informazioni, consulta [questo post del blog](#).

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 funzionante 1 o 2 esistente per iniziare a utilizzare l'SDK della versione 3. Una strategia di migrazione valida consiste nello scrivere solo nuovo codice sull'SDK della 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 root diverso, rispetto a. `Aws` `AWS` Ciò consente l'utilizzo. side-by-side
- `Aws.config`— Ora un hash Ruby alla vaniglia, invece di un metodo.
- Opzioni di costruzione rigorose: quando si costruisce un client o un oggetto risorsa nell'SDK della versione 1, le opzioni di costruzione sconosciute vengono ignorate. Nella versione 2, le opzioni del costruttore sconosciute attivano un. `ArgumentError` 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 client

Non ci sono differenze tra le classi di 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 client di servizi disporranno di interfacce compatibili dopo la costruzione del client. Alcune differenze importanti:

- `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` 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 di tipo vanilla.
- Rinomine 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 vengono rinominate nella versione 2. Altre vengono rimosse o sostituite. Ecco le principali modifiche:
  - `:use_ssl` è stato rimosso. La versione 2 utilizza SSL ovunque. Per disabilitare SSL è necessario configurare e utilizzare `:endpoint`. `http://`
  - `:ssl_ca_file` è adesso `:ssl_ca_bundle`
  - `:ssl_ca_path` è adesso `: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 di 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 la versione 2 le interfacce di risorse sono generate da un modello. Le interfacce di risorse della versione 2 sono notevolmente 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 getter 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 di batch della versione 2 sono molto diverse dalla versione 1.

# Sicurezza per AWS SDK for Ruby

La sicurezza cloud di Amazon Web Services (AWS) è la priorità più alta. In qualità di 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 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](#)
- [Identity and Access Management per AWS SDK for Ruby](#)
- [Convalida della conformità per AWS SDK for Ruby](#)
- [Resilienza per AWS SDK for Ruby](#)
- [Sicurezza dell'infrastruttura per AWS SDK for Ruby](#)
- [Applicazione di una versione TLS minima nell' AWS SDK for Ruby](#)
- [Migrazione del client di crittografia Amazon S3 \(da V1 a V2\)](#)
- [Migrazione del client di crittografia Amazon S3 \(da V2 a V3\)](#)

## Protezione dei dati in AWS SDK for Ruby

Il [modello di responsabilità AWS condivisa](#) 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. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i

Servizi AWS utilizzati. Per maggiori informazioni sulla privacy dei dati, consulta le [Domande frequenti sulla privacy dei dati](#). Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al [AWS Modello di responsabilità condivisa e GDPR](#) nel AWS Blog sulla sicurezza.

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. Sugeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta [Lavorare con i CloudTrail percorsi](#) nella Guida per l'AWS CloudTrail utente.
- 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 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il [Federal Information Processing Standard \(FIPS\) 140-3](#).

Ti consigliamo 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. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la 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 necessari un 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 gli specifici servizi Amazon Web Services (AWS) 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 gli specifici servizi Amazon Web Services (AWS) 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

- [Riferimenti sui servizi conformi ai requisiti HIPAA](#) - Elenca i servizi HIPAA idonei. Servizi AWS Non tutti sono idonei all'HIPAA.
- [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 gli specifici servizi Amazon Web 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 gli specifici servizi Amazon Web 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 consiste nell'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 (da V1 a V2)

### Note

Se utilizzi la versione 2 del client di crittografia S3 e desideri migrare alla versione 3, consulta [Migrazione del client di crittografia Amazon S3 \(da V2 a V3\)](#)

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, distribuisce 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 ne utilizza più AWS SDKs, è necessario aggiornare ogni SDK separatamente.
2. Migra i 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 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. 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. 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`

## Esegui la migrazione dei 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 un nuovo `EncryptionV2::Client`, devi fornire valori per `key_wrap_schema` e `content_encryption_schema`.

`key_wrap_schema`- Se si utilizza AWS KMS, questo deve essere impostato 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 `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, `EncryptionV2::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_allow_decrypt_with_any_cmk
kms_key_id: :kms_allow_decrypt_with_any_cmk:
true
get_object
```

## Esempio: utilizzo di una chiave simmetrica (AES)

### Pre-migrazione

```
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

### Pre-migrazione

```
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

### Pre-migrazione

```
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_cmek:
  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
```

## Migrazione del client di crittografia Amazon S3 (da V2 a V3)

### Note

Se si utilizza la versione 1 del client di crittografia S3, è necessario eseguire la migrazione alla versione 2 prima di passare alla versione 3. Consulta le istruzioni sulla [Migrazione del client di crittografia Amazon S3 \(da V1 a V2\)](#) migrazione dalla V1 alla V2.

Questo argomento mostra come migrare le applicazioni dalla versione 2 (V2) del client di crittografia Amazon Simple Storage Service (Amazon S3) alla versione 3 (V3) e garantire la disponibilità delle applicazioni durante tutto il processo di migrazione. V3 introduce AES GCM con Key Commitment and Commitment Policies per migliorare la sicurezza e proteggere dalla manomissione delle chiavi di dati.

## Panoramica sulla migrazione

La versione 3 del client di crittografia Amazon S3 introduce AES GCM con Key Commitment per una maggiore sicurezza. Questo nuovo algoritmo di crittografia fornisce protezione contro la manomissione delle chiavi dei dati e garantisce l'integrità dei dati crittografati. La migrazione alla V3 richiede un'attenta pianificazione per mantenere la disponibilità delle applicazioni e l'accessibilità dei dati durante l'intero processo.

Questa migrazione avviene in due fasi:

1. Aggiorna i client esistenti per leggere nuovi formati. Innanzitutto, distribuisce una versione aggiornata dell' AWS SDK for Ruby nella tua applicazione. Ciò consentirà ai client di crittografia V2 esistenti di decrittografare gli oggetti scritti dai nuovi client V3. Se l'applicazione ne utilizza più AWS SDKs, è necessario aggiornare ogni SDK separatamente.

2. Migra i client di crittografia e decrittografia alla V3. Una volta che tutti i client di crittografia V2 saranno in grado di leggere nuovi formati, sarà possibile migrare i client di crittografia e decrittografia esistenti alle rispettive versioni V3. Ciò include la configurazione delle politiche di impegno e l'aggiornamento del codice per utilizzare le nuove opzioni di configurazione del client.

Se non hai ancora effettuato la migrazione dalla V1 alla V2, devi prima completare la migrazione. Consulta [Migrazione del client di crittografia Amazon S3 \(da V1 a V2\)](#) le istruzioni dettagliate sulla migrazione dalla V1 alla V2.

## Comprensione delle funzionalità della V3

La versione 3 del client di crittografia Amazon S3 introduce due funzionalità di sicurezza chiave: Commitment Policies e AES GCM con Key Commitment. La comprensione di queste funzionalità è essenziale per pianificare la strategia di migrazione e garantire la sicurezza dei dati crittografati.

### Politiche di impegno

Le politiche di impegno controllano il modo in cui il client di crittografia gestisce l'impegno delle chiavi durante le operazioni di crittografia e decrittografia. Key Commitment garantisce che i dati crittografati possano essere decrittografati solo con la chiave esatta utilizzata per crittografarli, proteggendoli da determinati tipi di attacchi crittografici.

Il client di crittografia V3 supporta tre opzioni di Commitment Policy:

#### **FORBID\_ENCRYPT\_ALLOW\_DECRYPT**

Questa politica crittografica gli oggetti senza impegno di chiave e consente la decrittografia di entrambi gli oggetti con e senza impegno chiave.

- Comportamento di crittografia: gli oggetti vengono crittografati senza impegno di chiave, utilizzando la stessa suite di algoritmi della versione V2.
- Comportamento di decrittografia: può decrittografare oggetti crittografati con o senza impegno di chiave.
- Implicazioni sulla sicurezza: questa politica non impone l'impegno fondamentale e può consentire la manomissione. Gli oggetti crittografati con questa politica non beneficiano delle protezioni di

sicurezza avanzate di Key Commitment. Utilizzate questa policy solo durante la migrazione quando è necessario mantenere la compatibilità con il comportamento di crittografia V2.

- Compatibilità delle versioni: gli oggetti crittografati con questa politica possono essere letti da tutte le implementazioni V2 e V3 del client di crittografia S3.

## **REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT**

Questa policy crittografa gli oggetti con impegno chiave e consente la decrittografia di entrambi gli oggetti con e senza impegno chiave.

- Comportamento di crittografia: gli oggetti vengono crittografati con impegno chiave utilizzando AES GCM con Key Commitment.
- Comportamento di decrittografia: può decrittografare oggetti crittografati con o senza impegno chiave, garantendo la compatibilità con le versioni precedenti.
- Implicazioni sulla sicurezza: i nuovi oggetti traggono vantaggio dalla protezione con impegno chiave, mentre gli oggetti esistenti senza impegno chiave possono ancora essere letti. Ciò fornisce un equilibrio tra sicurezza e compatibilità con le versioni precedenti durante la migrazione.
- Compatibilità delle versioni: gli oggetti crittografati con questa politica possono essere letti solo dalle implementazioni V3 e V2 più recenti del client di crittografia S3.

## **REQUIRE\_ENCRYPT\_REQUIRE\_DECRYPT**

Questa politica crittografa gli oggetti con impegno chiave e consente la decrittografia solo degli oggetti che sono stati crittografati con impegno chiave.

- Comportamento di crittografia: gli oggetti vengono crittografati con impegno chiave utilizzando AES GCM con Key Commitment.
- Comportamento di decrittografia: può decrittografare solo gli oggetti che sono stati crittografati con l'impegno della chiave. I tentativi di decrittografare gli oggetti senza l'impegno di una chiave falliranno.
- Implicazioni sulla sicurezza: questa politica offre il massimo livello di sicurezza imponendo un impegno fondamentale per tutte le operazioni. Utilizzate questa policy solo dopo che tutti gli oggetti sono stati ricrittografati con key commitment e tutti i client sono stati aggiornati alla V3.
- Compatibilità delle versioni: gli oggetti crittografati con questa policy possono essere letti solo dalle implementazioni V3 e V2 più recenti del client di crittografia S3. Questa politica impedisce inoltre la lettura di oggetti crittografati dai client V2 o V1.

**Note**

Quando pianificate la migrazione, iniziate `REQUIRE_ENCRYPT_ALLOW_DECRYPT` a mantenere la compatibilità con le versioni precedenti, ottenendo al contempo i vantaggi in termini di sicurezza derivanti dall'impegno chiave per i nuovi oggetti. Passa a questa versione solo `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` dopo che tutti gli oggetti sono stati ricrittografati e tutti i client sono stati aggiornati alla V3.

## AES GCM con impegno chiave

AES GCM with Key Commitment (`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`) è un nuovo algoritmo di crittografia introdotto nella V3 che offre una maggiore sicurezza proteggendo dalla manomissione delle chiavi di dati. Comprendere come funziona questo algoritmo e quando viene applicato è importante per pianificare la migrazione.

In che modo `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` si differenzia dagli algoritmi precedenti

Le versioni precedenti del client di crittografia S3 utilizzavano AES CBC o AES GCM senza impegno chiave per crittografare la chiave dei dati nei file di istruzioni. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` aggiunge un impegno crittografico al processo di crittografia, che associa i dati crittografati a una chiave specifica. Ciò impedisce a un utente malintenzionato di manomettere la chiave di dati crittografata nel file di istruzioni e di indurre il client a decrittografare i dati con una chiave errata.

Senza un impegno chiave, un utente malintenzionato può modificare la chiave di dati crittografata in un file di istruzioni in modo che venga decrittografata su una chiave diversa, permettendo potenzialmente l'accesso non autorizzato o il danneggiamento dei dati. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` previene questo attacco assicurando che la chiave dati crittografata possa essere decrittografata solo sulla chiave originale utilizzata durante la crittografia.

### Compatibilità delle versioni

Gli oggetti crittografati con `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` possono essere decrittografati solo dalle implementazioni V3 del client di crittografia S3 e da alcune versioni di transizione di V2 che includono il supporto per la lettura dei formati V3. I client V2 senza

questo supporto per la transizione non possono decrittografare i file di istruzioni crittografati con `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

### Warning

Prima di abilitare la crittografia con `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` (utilizzando `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` impegnando le politiche), assicuratevi che tutti i client che devono leggere gli oggetti crittografati siano stati aggiornati alla V3 o a una versione di transizione che supporti i formati V3. Se un client V2 senza supporto di transizione tenta di leggere oggetti crittografati con `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`, la decrittografia avrà esito negativo.

Durante la migrazione, è possibile utilizzare la policy di commit per continuare a crittografare senza consentire `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` comunque ai client V3 di leggere gli oggetti crittografati con l'impegno della chiave `FORBID_ENCRYPT_ALLOW_DECRYPT`. Ciò fornisce un percorso di migrazione sicuro in cui si aggiornano prima tutti i lettori, quindi si passa alla crittografia con impegno chiave.

## Aggiorna i client esistenti per leggere nuovi formati

Il client di crittografia V3 utilizza algoritmi di crittografia e funzionalità di impegno chiave che i client V2 non supportano di default. Il primo passo della migrazione consiste nell'aggiornare i client di decrittografia V2 a una versione dell'SDK for AWS Ruby in grado di leggere gli oggetti crittografati V3. Dopo aver completato questo passaggio, i client V2 dell'applicazione saranno in grado di decrittografare gli oggetti crittografati dai client di crittografia V3.

Per leggere gli oggetti crittografati dai client V3 (quelli che utilizzano `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o rispettano le politiche `REQUIRE_ENCRYPT_REQUIRE_DECRYPT`), è necessario utilizzare la versione 1.93.0 o successiva del gem. `aws-sdk-s3` Questa versione include il supporto per la decrittografia degli oggetti crittografati con AES GCM con Key Commitment.

Installazione dalla riga di comando

Per i progetti che installano il `aws-sdk-s3` gem dalla riga di comando, usa l'opzione `version` per verificare che sia installata la versione minima di 1.208.0.

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

## Usare Gemfiles

Per i progetti che utilizzano un Gemfile per gestire le dipendenze, imposta la versione minima del `aws-sdk-s3` gem su 1.208.0. Esempio:

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. Modifica il tuo Gemfile per specificare la versione minima.
2. Esegui `bundle update aws-sdk-s3` per aggiornare la gemma.
3. Per verificare la tua versione, `bundle info aws-sdk-s3` esegui.

### Note

Dopo l'aggiornamento alla versione più recente, i client di crittografia V2 esistenti saranno in grado di decrittografare gli oggetti crittografati dai client V3. Tuttavia, continueranno a crittografare nuovi oggetti utilizzando algoritmi V2 fino a quando non li migrerai alla V3 come descritto nella sezione successiva.

## Migrazione dei client di crittografia e decrittografia alla V3

Dopo aver aggiornato i client per leggere i nuovi formati di crittografia, è possibile aggiornare le applicazioni ai client di crittografia e decrittografia V3. I passaggi seguenti mostrano come migrare correttamente il codice da V2 a V3.

Prima di aggiornare il codice per utilizzare il client di crittografia V3, assicurati di aver seguito i passaggi precedenti e di utilizzare la versione `aws-sdk-s3` gem 1.93.0 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 V3

Il client di crittografia V3 introduce nuove opzioni di configurazione che controllano il comportamento degli impegni chiave e la compatibilità con le versioni precedenti. La comprensione di queste opzioni è essenziale per una migrazione di successo.

### politica\_di impegno

Il `commitment_policy` parametro controlla il modo in cui il client di crittografia gestisce l'impegno delle chiavi durante le operazioni di crittografia e decrittografia. Questa è l'opzione di configurazione più importante per i client V3.

- `:require_encrypt_allow_decrypt`- Crittografia nuovi oggetti con impegno chiave e consente la decrittografia degli oggetti con o senza impegno chiave. Questa è l'impostazione consigliata per la migrazione, in quanto offre una maggiore sicurezza per i nuovi oggetti mantenendo al contempo la compatibilità con le versioni precedenti con gli oggetti V2 esistenti.
- `:forbid_encrypt_allow_decrypt`- Crittografia nuovi oggetti senza impegno chiave (utilizzando algoritmi V2) e consente la decrittografia degli oggetti con o senza impegno chiave. Utilizzate questa impostazione solo se è necessario mantenere il comportamento di crittografia V2 durante la migrazione, ad esempio quando alcuni client non sono ancora in grado di leggere gli oggetti crittografati V3.
- `:require_encrypt_require_decrypt`- Crittografia i nuovi oggetti con impegno chiave e consente la decrittografia solo degli oggetti che sono stati crittografati con impegno chiave. Utilizzate questa impostazione solo dopo che tutti gli oggetti sono stati ricrittografati con key commitment e tutti i client sono stati aggiornati alla V3.

### profilo\_sicurezza

Il `security_profile` parametro determina il supporto per la lettura di oggetti scritti da versioni precedenti dei client di crittografia. Questo parametro è essenziale per mantenere la compatibilità con le versioni precedenti durante la migrazione.

- `:v3_and_legacy`- Consente al client V3 di decrittografare gli oggetti crittografati dai client di crittografia V1 e V2. Utilizzate questa impostazione durante la migrazione per assicurarvi che i client V3 possano leggere tutti gli oggetti crittografati esistenti.
- `:v3`- Consente al client V3 di decrittografare gli oggetti crittografati solo dai client di crittografia V2. Utilizzate questa impostazione se avete già migrato tutti gli oggetti V1 al formato V2.

- Se non specificato, il client decrittograferà solo gli oggetti crittografati dai client V3. Utilizzatelo solo per lo sviluppo di nuove applicazioni in cui non esistono oggetti legacy.

## envelope\_location

Il `envelope_location` parametro determina dove vengono archiviati i metadati di crittografia (inclusa la chiave di dati crittografata). Questo parametro influisce sugli oggetti protetti da AES GCM con Key Commitment.

- `:metadata`(Impostazione predefinita): archivia i metadati di crittografia nelle intestazioni dei metadati dell'oggetto S3. Questo è il comportamento predefinito ed è consigliato per la maggior parte dei casi d'uso. Quando si utilizza l'archiviazione dei metadati, non si applica AES GCM con Key Commitment.
- `:instruction_file`- Memorizza i metadati di crittografia in un oggetto S3 separato (file di istruzioni) con un suffisso configurabile. Quando si utilizzano i file di istruzioni, AES GCM con Key Commitment protegge la chiave dati crittografata dalla manomissione. Utilizza questa impostazione se hai bisogno della sicurezza aggiuntiva fornita da Key Commitment per la chiave dati stessa.

Quando si utilizza `:instruction_file`, è possibile specificare facoltativamente il `instruction_file_suffix` parametro per personalizzare il suffisso utilizzato per gli oggetti Instruction File. Il suffisso predefinito è `.instruction`

Quando utilizzare ciascuna opzione di configurazione

Durante la migrazione, seguite questa strategia di configurazione consigliata:

1. Migrazione iniziale: imposta `commitment_policy: :require_encrypt_allow_decrypt` e `security_profile: :v3_and_legacy`. Ciò consente ai client V3 di crittografare nuovi oggetti con impegno chiave, pur essendo in grado di decrittografare tutti gli oggetti V1 e V2 esistenti.
2. Dopo l'aggiornamento di tutti i client: continua a utilizzare `commitment_policy: :require_encrypt_allow_decrypt` e `security_profile: :v3_and_legacy` fino a quando non avrai ricrittografato tutti gli oggetti che richiedono una protezione con impegno chiave.
3. Applicazione V3 completa: solo dopo che tutti gli oggetti sono stati ricrittografati con key commit e non è più necessario leggere gli oggetti V1/V2, è possibile facoltativamente passare al `security_profile` parametro

`commitment_policy`: `:require_encrypt_require_decrypt` e rimuoverlo (o impostarlo su se gli oggetti V2 esistono ancora). `:v2`

Ad esempio `envelope_location`, continuate a utilizzare il metodo di archiviazione esistente (`:metadatao:instruction_file`) a meno che non abbiate un motivo specifico per modificarlo. Se attualmente utilizzi l'archiviazione dei metadati e desideri la sicurezza aggiuntiva di AES GCM con Key Commitment per la chiave dati, puoi passare a `:instruction_file`, ma tieni presente che ciò richiederà l'aggiornamento di tutti i client che leggono questi oggetti.

## Esegui la migrazione dei client di crittografia e decrittografia alla V3

Dopo aver aggiornato i client per leggere i nuovi formati di crittografia, è possibile aggiornare le applicazioni ai client di crittografia e decrittografia V3. Gli esempi seguenti mostrano come migrare correttamente il codice da V2 a V3.

### Utilizzo dei client di crittografia V3

#### Premigrazione (V2)

```
require 'aws-sdk-s3'

# Create V2 encryption client with KMS
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy,
  commitment_policy: :forbid_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

#### Durante la migrazione (V3 con compatibilità con le versioni precedenti)

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

```
# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

## Dopo la migrazione (V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3,
  # Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
  # This encrypts with key commitment and does not decrypt V2 objects
  commitment_policy: :require_encrypt_require_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

La differenza fondamentale in V3 è l'aggiunta del parametro `commitment_policy` impostandolo in `:require_encrypt_require_decrypt` modo da garantire che i nuovi oggetti siano crittografati con l'impegno della chiave e che il client decifri solo gli oggetti crittografati con l'impegno della chiave, garantendo una maggiore sicurezza contro la manomissione delle chiavi di dati.

La `put_object` chiamata stessa rimane invariata. Tutti i miglioramenti della sicurezza sono configurati a livello di client.

## Esempi aggiuntivi

Questa sezione fornisce esempi aggiuntivi per scenari di migrazione e opzioni di configurazione specifici che possono essere utili durante la migrazione da V2 a V3.

### File di istruzioni e archiviazione dei metadati

Il client di crittografia S3 può archiviare i metadati di crittografia (inclusa la chiave dei dati crittografati) in due posizioni diverse: nelle intestazioni dei metadati dell'oggetto S3 o in un file di istruzioni separato. La scelta del metodo di archiviazione influisce sugli oggetti che beneficiano della protezione AES GCM con Key Commitment.

### Archiviazione dei metadati (impostazione predefinita)

Per impostazione predefinita, il client di crittografia archivia i metadati di crittografia nelle intestazioni dei metadati dell'oggetto S3. Questo è l'approccio consigliato per la maggior parte dei casi d'uso perché mantiene i metadati di crittografia con l'oggetto e non richiede la gestione di oggetti separati del file di istruzioni.

```
require 'aws-sdk-s3'

# Create V3 encryption client with metadata storage (default)
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :metadata # Explicitly set to metadata (this is the default)
)

# Encrypt and upload object
# Encryption metadata is stored in the object's metadata headers
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')
```

Quando si utilizza l'archiviazione dei metadati, AES GCM con Key Commitment non si applica alla chiave dati crittografata. Tuttavia, la crittografia dei contenuti beneficia ancora dell'impegno chiave quando si utilizza `commitment_policy: :require_encrypt_allow_decrypt` o `:require_encrypt_require_decrypt`.

## Archiviazione dei file di istruzioni

In alternativa, puoi configurare il client di crittografia per archiviare i metadati di crittografia in un oggetto S3 separato chiamato file di istruzioni. Quando si utilizzano Instruction Files with V3, la chiave dati crittografata è protetta da AES GCM con Key Commitment, che fornisce una sicurezza aggiuntiva contro la manomissione delle chiavi di dati.

```
require 'aws-sdk-s3'

# Create V3 encryption client with instruction file storage
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :instruction_file, # Store metadata in separate instruction file
  instruction_file_suffix: '.instruction' # Optional: customize the suffix (default is
'.instruction')
)

# Encrypt and upload object
# Encryption metadata is stored in a separate object: 'my-object.instruction'
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# When retrieving the object, the client automatically reads the instruction file
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```


Quando viene utilizzato `envelope_location: :instruction_file`, il client di crittografia crea due oggetti S3:

1. L'oggetto di dati crittografato (ad esempio, `my-object`)
2. Il file di istruzioni contenente i metadati di crittografia (ad es.) `my-object.instruction`

Il `instruction_file_suffix` parametro consente di personalizzare il suffisso utilizzato per i file di istruzioni. Il valore predefinito è `.instruction`.

Quando utilizzare ciascun metodo di archiviazione

- Utilizza l'archiviazione dei metadati per la maggior parte degli scenari. Semplifica la gestione degli oggetti poiché i metadati di crittografia viaggiano con l'oggetto.
- Utilizzate Instruction File Storage quando la dimensione dei metadati degli oggetti è un problema o quando è necessario separare i metadati di crittografia dall'oggetto crittografato. Tieni presente che l'utilizzo dei file di istruzioni richiede la gestione di due oggetti S3 (l'oggetto crittografato e il relativo file di istruzioni) anziché uno.

 Warning

Se passi dalla memorizzazione dei metadati alla memorizzazione dei file di istruzioni (o viceversa), gli oggetti esistenti crittografati con il vecchio metodo di archiviazione non saranno leggibili dai client configurati con il nuovo metodo di archiviazione. Pianificate attentamente il metodo di archiviazione e mantenete la coerenza in tutta l'applicazione.

# Cronologia dei documenti

La tabella seguente descrive le modifiche importanti di questa guida. Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile [isciversi a un feed RSS](#).

Modifica	Descrizione	Data
<a href="#">Riorganizzazione dei contenuti</a>	Aggiornamento del sommario e dell'organizzazione dei contenuti per allinearli meglio agli altri. AWS SDKs	29 marzo 2025
<a href="#">Osservabilità</a>	Sono state aggiunte informazioni relative all'osservabilità di Ruby.	24 gennaio 2025
<a href="#">Aggiornamenti generali</a>	Versione Ruby minima richiesta aggiornata alla v2.5. Collegamenti alle risorse aggiornati.	13 novembre 2024
<a href="#">Sommario ed esempi guidati</a>	Esempi guidati rimossi per rimandare al più completo Code Examples Repository.	10 luglio 2024
<a href="#">Sommario</a>	Sommario aggiornato per rendere più accessibili gli esempi di codice.	1 giugno 2023
<a href="#">Aggiornamenti delle best practice di IAM</a>	Guida aggiornata per l'allineamento alle best practice IAM. Per ulteriori informazioni, consulta <a href="#">Best practice per la sicurezza in IAM</a> . Aggiornamenti alla Guida introduttiva.	08 maggio 2023
<a href="#">Aggiornamenti generali</a>	Aggiornamento della pagina di benvenuto per le risorse	8 agosto 2022

esterne pertinenti. È stata inoltre aggiornata la versione minima richiesta di Ruby per la v2.3. AWS Key Management Service Sezioni aggiornate per riflettere gli aggiornamenti terminologici. Informazioni di utilizzo aggiornate sull'utilità REPL per maggiore chiarezza.

### [Correzione dei collegamenti interrotti](#)

Sono stati corretti gli esempi di link non funzionanti. È stata rimossa la pagina Tips and Tricks ridondante; reindirizzamento a contenuti di esempio di Amazon EC2 . Elenchi inclusi degli esempi di codice disponibili GitHub nel repository Code Examples.

3 agosto 2022

### [SDK Metrics](#)

Sono state rimosse le informazioni sull'attivazione di SDK Metrics for Enterprise Support, che sono diventate 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.